ALIBABA CLOUD

Alibaba Cloud

Container Service for Kubernetes

User Guide for Kubernetes Clusters

Document Version: 20210713

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Overview

Kubernetes is the mainstream open source platform for container orchestration. Alibaba Cloud provides Container Service for Kubernetes (ACK). This service allows users to create ACK clusters and manage containerized applications in ACK clusters.
You can log on to the ACK console to create ACK clusters that support high security and high availability. ACK clusters integrate the virtualization, storage, networking, and security capabilities of Alibaba Cloud. ACK clusters are high-performance and scalable. You can use ACK clusters to manage containerized applications. In addition, you can create or expand ACK clusters with a few steps. This allows you to focus on the work of developing and managing containerized applications.

Alibaba Cloud provides multiple types of ACK clusters to meet the requirements of diverse scenarios.

- Managed Kubernetes clusters are most commonly used and can be applied in most scenarios.
- Serverless Kubernetes clusters are applicable to scenarios where agility is important and can be used to process individual and multiple tasks.
- Edge Kubernetes clusters are the most suitable option when you want to handle edge services such as Internet of Things (IoT) and Content Delivery Network (CDN).
- Managed Kubernetes clusters for confidential computing are the most suitable option for business that requires high data security.

ACK also provides solutions to Alibaba Cloud Genomics Service (AGS) and AI-empowered big data computing for tight service integrations. ACK maximizes container performance by using the computing and networking capabilities of Infrastructure-as-a-Service (IaaS). ACK allows you to centrally manage more than one cluster deployed in multi-cloud and hybrid cloud environments. You can log on to the ACK console to manage your Kubernetes clusters deployed in on-premises data centers or other clouds.

ACK supports full lifecycle management for containerized applications and provides the following features:

- Cluster management
  Allows you to create ACK clusters with a few steps in the console. You can deploy ACK clusters across zones for high availability.
- All-in-one container management
  - Supports full lifecycle management for containerized applications.
  - Supports the Flannel and Terway network plug-ins.
  - Allows you to persist data to cloud disks, Network Attached Storage (NAS) file systems, and Object Storage Service (OSS) buckets provided by Alibaba Cloud.
  - Monitors resources, applications, and containers.
  - Provides diverse methods to collect log data and generate reports.
  - Supports Role-Based Access Control (RBAC) and ensures the security of container runtimes.
- Application marketplace
  - Provides rich Helm chart components.
  - Integrates the Container Registry service.
- Developer services
  - Supports the OpenAPI specification and APIs developed by the Kubernetes community.
  - Supports Cloud Shell developed by Alibaba Cloud and the Kubectl tool developed by the Kubernetes community.

Related open source projects

Open source projects of ACK: Aliyun Container Service.

If you have any questions or suggestions for related projects, raise an issue or pull a request in the Kubernetes community.

Service level agreement (SLA)

For more information, see Container Service for Kubernetes Service-level Agreement.

2.Authorization management

2.1. Authorization overview

The authorization mechanism of Container Service for Kubernetes (ACK) consists of Resource Access Management (RAM) authorization and role-based access control (RBAC) authorization. This topic describes RAM authorization and RBAC authorization and how to use these authorizations.

RAM authorization

In scenarios where RAM is integrated with enterprise account systems, operations and maintenance (O&M) engineers frequently manage cloud resources as RAM users. By default, a RAM user is not authorized to call the APIs of cloud resources. To allow a RAM user to call the API, you must grant the required permissions to the RAM user.

If you want to scale a cluster, add nodes to a cluster, or access the cluster as a RAM user, you must grant the required permissions to the RAM user. For more information, see Create a custom RAM policy.

You can perform RAM authorization by using the following methods:

- Attach system policies: You can use this method to grant a RAM user read and write permissions on all clusters that belong to the current Alibaba Cloud account. To grant a RAM user the permissions to manage all clusters that belong to the current Alibaba Cloud account, we recommend that you attach system policies to the RAM user. For more information about how to attach system policies to a RAM user, see Grant permissions to a RAM user.
  The following table describes the commonly used system policies for ACK. You can select the system policies based on your requirements.

System policy	Description
AliyunCSFullAccess	Allows a RAM user to fully control all ACK clusters that belong to the current Alibaba Cloud account.
AliyunVPCReadOnlyAccess	Allows a RAM user to specify a virtual private cloud (VPC) when the RAM user creates a cluster.
AliyunECSReadOnlyAccess	Allows a RAM user to add existing nodes to a specified cluster and view the details of nodes.
AliyunContainerRegistryFullAccess	Allows a RAM user to fully control the images of all workloads that belong to the current Alibaba Cloud account.
AliyunLogReadOnlyAccess	Allows a RAM user to specify an existing Log Service project to store auditing logs when the RAM user creates a cluster and view the inspection details of a specified cluster.
AliyunAHASReadOnlyAccess	Allows a RAM user to enable the cluster topology feature.
AliyunRAMFullAccess	Allows a RAM user to manage the authorization of all RAM users of the current Alibaba Cloud account.
AliyunYundunSASReadOnlyAccess	Allows a RAM user to view the runtime monitoring data of a specified cluster.

System policy	Description
AliyunARMSReadOnlyAccess	Allows a RAM user to view the Prometheus monitoring state of a specified cluster.
AliyunKMSReadOnlyAccess	Allows a RAM user to enable Secret encryption when the RAM user creates a professional managed Kubernetes cluster.

- Attach custom policies: You can use this method to implement fine-grained access control on cloud resources for a RAM user. If a RAM user requires permissions for custom development by using SDKs, you can use this method to grant the RAM user the permissions to call specified API operations.
  For more information about how to attach custom policies to a RAM user for cluster-level access control, see Create a custom RAM policy.
  For example, if a RAM user requires read permissions on a specified Object Storage Service (OSS) bucket, you can create a custom policy based on the following content and attach it to the RAM user.

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
            "oss:ListBuckets",
            "oss:GetBucketStat",
            "oss:GetBucketInfo",
            "oss:GetBucketTagging",
            "oss:GetBucketAcl"
            ],
      "Resource": "acs:oss:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "oss:ListObjects",
        "oss:GetBucketAcl"
      ],
      "Resource": "acs:oss:*:*:myphotos"
    },
    {
      "Effect": "Allow",
      "Action": [
        "oss:GetObject",
        "oss:GetObjectAcl"
      ],
      "Resource": "acs:oss:*:*:myphotos/*"
    }
  ]
}
```

RBAC authorization

If a RAM user requires permissions to manage Kubernetes resources in a specified cluster, you must go to the Authorizations page of the Container Service for Kubernetes (ACK) console console and grant the RAM user resource-level permissions, such as the permissions to view information about pods and nodes.

You can assign the following predefined roles to a RAM user. For more information, see Assign RBAC roles to a RAM user.

Roles and permissions

Role	RBAC permissions on cluster resources
Administrator	Read and write permissions on resources in all namespaces.
O&M Engineer	Read and write permissions on visible resources in the console in all namespaces and read-only permissions on nodes, persistent volumes (PVs), namespaces, and quotas.
Developer	Read and write permissions on visible resources in the console in a specified namespace or all namespaces.
Restricted User	Read-only permission on visible resources in the console in a specified namespace or all namespaces.
Custom role	The permissions of a custom role are determined by the ClusterRole that you select. Before you select a ClusterRole, check the permissions of the ClusterRole and make sure that you grant only the required permissions to the RAM user.

Related information

- FAQ about authorization management
- ACK default roles
- The service linked role for ACK

2.2. Assign RBAC roles to a RAM user

This topic describes how to assign role-based access control (RBAC) roles to Resource Access Management (RAM) users.

Prerequisites

- By default, RAM users that are not the creators of clusters do not have access to the resources in Container Service for Kubernetes (ACK) clusters.
- To assign RBAC roles to a RAM user, you must make sure that the RAM user is granted at least read-only permissions on specified clusters in the RAM console.
- ACK provides two predefined roles, Administrator and O&M Engineer. The Administrator role has full access to all of the resources in the clusters. A RAM user can assume one of these roles to grant permissions to others RAM users.
- If the RAM user assumes the Administrator role, the RAM user can grant other RAM users all cluster-scoped permissions. Newly created clusters are automatically bound to ClusterRole.
- When the RAM user assumes the O&M Engineer role to authorize other RAM users, only the clusters and namespaces that the current RAM user can manage are listed in the console. In addition, the RAM user must be assigned the Administrator or cluster-admin role of the specified cluster or namespace. Otherwise, the RAM user is not allowed to authorize other RAM users to access the specified cluster or namespace.
- You can assign RBAC roles to multiple RAM users at a time.
- To ensure data security, you are not allowed to modify RAM permission policies that are attached to your RAM users in the ACK console. You must read the instructions on the authorization page, log on to the RAM console, and modify the permission policies.

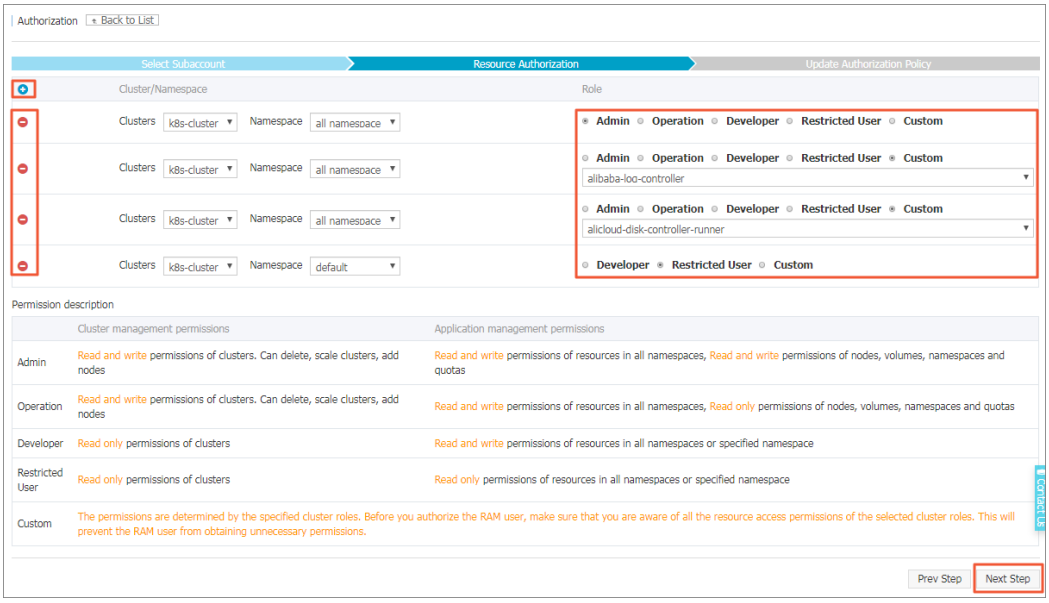
Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Authorizations** to go to the Authorizations page.
3. On the **Authorizations** page, click the **RAM Users** tab. On the RAM Users tab, find the RAM user to which you want to grant permissions and click **Modify Permissions** to open the **Configure Role-Based Access Control (RBAC)** wizard page.

   > **Note** If you log on to the ACK console as a RAM user, make sure that the RAM user has at least read-only permissions on the cluster that you want to manage. In addition, the RAM user must be assigned the cluster-admin role or predefined RBAC administrator role. For more information, see Customize RAM permissions

4. On the **Configure Role-Based Access Control (RBAC)** wizard page, click the plus sign (+) to add cluster-scoped or namespace-scoped permissions and select a predefined or custom RBAC role in the Permission column. You can also click the minus sign (-) to delete permissions. After you add the permissions, click **Next Step**.

   > **Note** For each RAM user, you can assign only one predefined RBAC role but one or more custom RBAC roles to manage the same cluster or namespace.



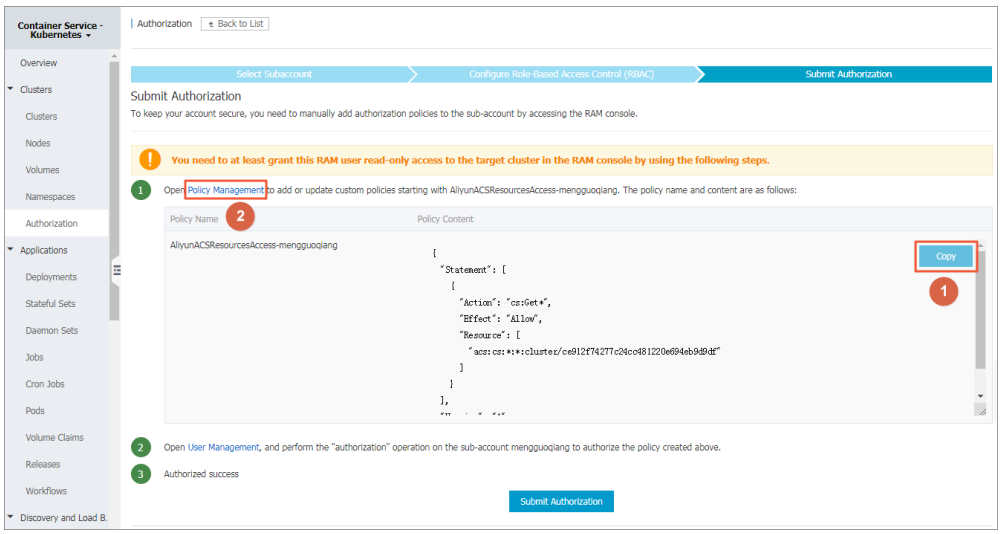
The following table describes permissions that the predefined and custom RBAC roles have on clusters and namespaces.
Roles and permissions

Role	RBAC permissions on cluster resources
Administrator	Read and write permissions on resources in all namespaces.
O&M Engineer	Read and write permissions on resources in all namespaces and read-only permissions on nodes, persistent volumes (PVs), namespaces, and service quotas within a cluster.
Developer	Read and write permissions on resources in a specified namespace or all namespaces.

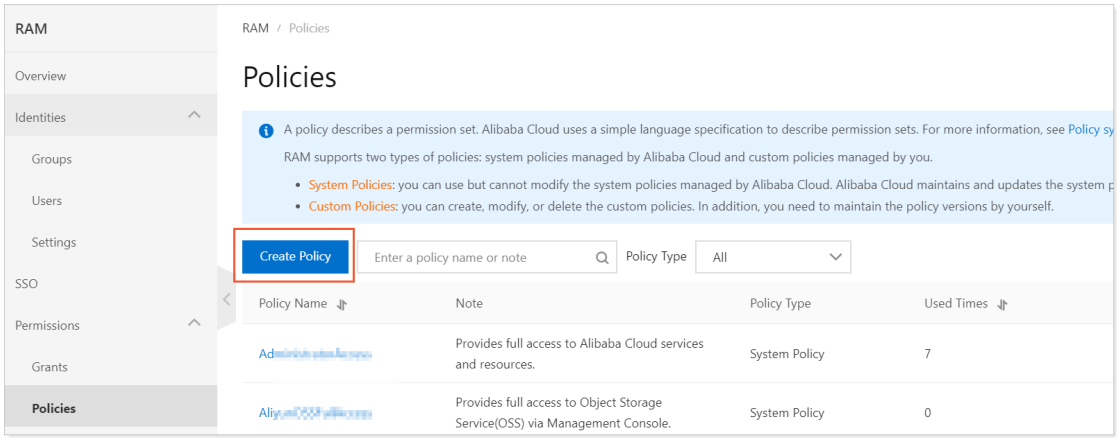
Role	RBAC permissions on cluster resources
Restricted User	Read-only permissions on resources in a specified namespace or all namespaces.
Custom	The cluster role that you select for a custom role determines what permissions the custom role has. Before you select a cluster role, make sure that you are aware of the permissions that the cluster role has in case the RAM user is granted excessive permissions. For more information, see Customize RAM permission policies.

5. On the **Submit Authorization** wizard page, if **The authorization is complete** appears, it indicates that the RBAC roles are assigned to the RAM user. If you see the result in the following Authorizations figure, it indicates that RBAC roles are not assigned because the RAM user is not authorized to manage the cluster. You must read the instructions on the authorizations page, log on to the RAM console, and then grant the RAM user at least read-only permissions on the cluster.
   i. On the **Submit Authorization** wizard page, click **Copy** and then click **policy management** to go to the RAM console.

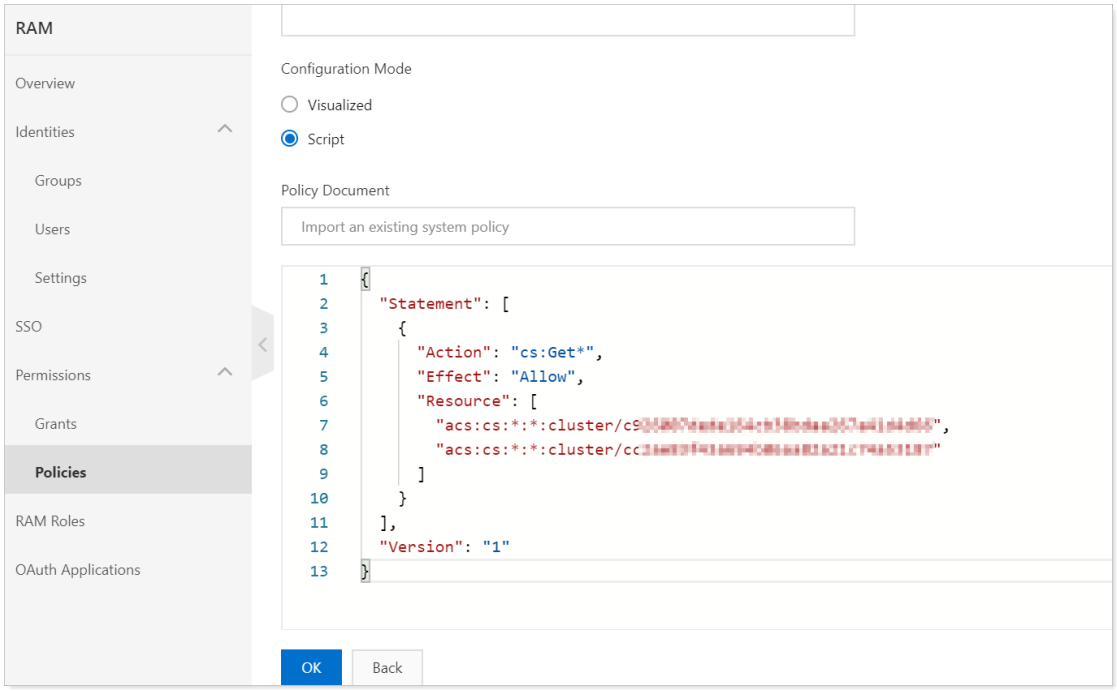
      Authorizations



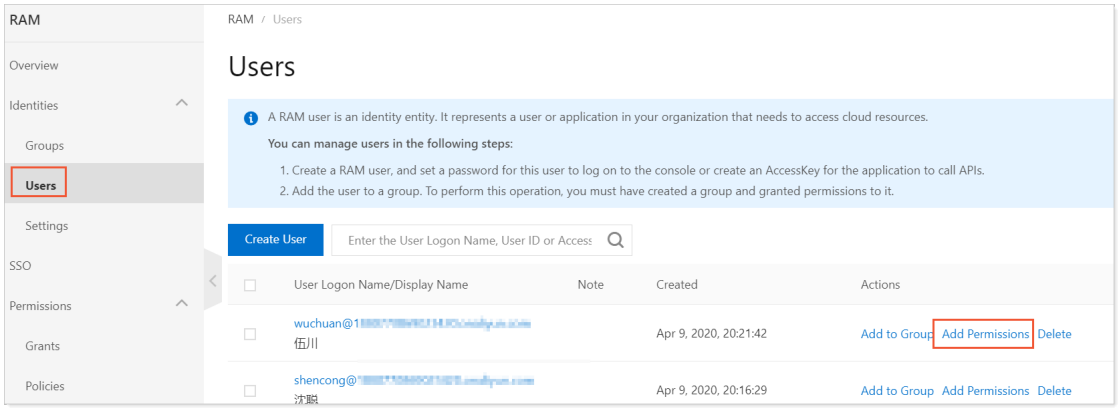
   ii. In the left-side navigation pane, choose **Permissions > Policies**. On the Policies page, click **Create Policy**.



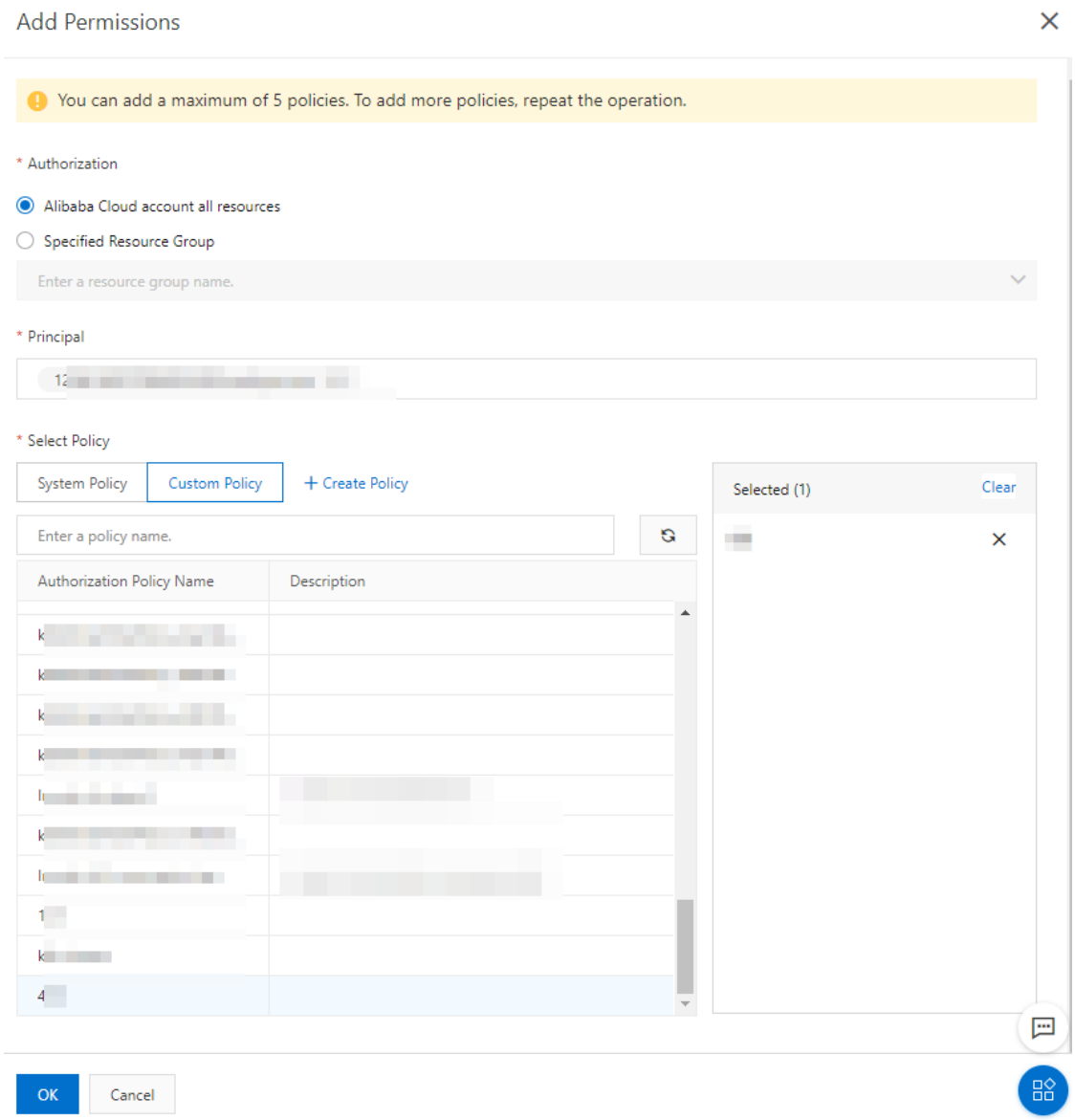
iii. On the **Create Custom Policy** page, enter a **name** for the policy and set Configuration Mode to **Script**. Then, press **Ctrl+V** to paste the policy content that was copied in Step 5.a to the **Policy Document** section, and click **OK**. For more information, see 自定义RAM授权策略.



iv. In the left-side navigation pane of the RAM console, choose **Identities > Users**. On the Users page, find the RAM user to which you want to grant permissions and click **Add Permissions** in the Actions column.



v. In the **Add Permissions** pane, set **Authorization**, select **Custom Policy**, and find and click the name of the policy that was created in Step 5.c. The policy is added to the **Selected** section on the right side of the page. Click **OK**. Read-only permissions on the specified cluster are granted to the RAM user.



vi. Return to the Container Service console. On the **Submit Authorization** wizard page, click **Submit Authorization**. The RBAC roles are assigned to the RAM user.

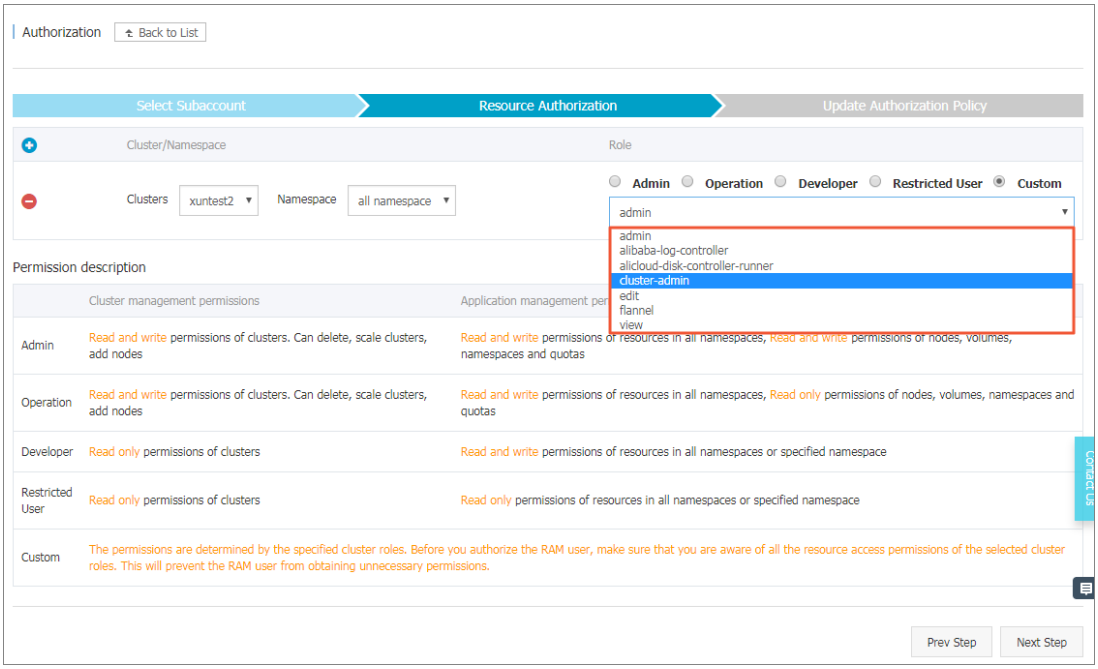
6. After the authorization is complete, you can log on to the ACK console as the authorized RAM user and perform the allowed operations to manage ACK.

Predefined and custom RBAC roles

ACK provides the following predefined RBAC roles: administrator, O&M engineer, developer, and restricted user. You can use these roles to regulate ACK access control in most scenarios. In addition, you can use custom roles to customize permissions on clusters.

ACK provides a set of custom RBAC roles.

Note The cluster-admin role is similar to a super administrator. By default, the cluster-admin role has the permissions to manage all resources within a cluster.



You can log on to a master node of a cluster and run the following command to view the custom RBAC roles that are assigned to the current account:

```
kubectl get clusterrole
```

```
kubectl get clusterrole
NAME                                   AGE
admin                                  13d
alibaba-log-controller                 13d
alicloud-disk-controller-runner        13d
cluster-admin                          13d
cs:admin                               13d
edit                                   13d
flannel                                13d
kube-state-metrics                     22h
node-exporter                          22h
prometheus-k8s                         22h
prometheus-operator                    22h
system:aggregate-to-admin              13d
....
system:volume-scheduler                13d
view                                   13d
```

Run the following command to view the details of a role, for example, the cluster-admin role:

```
kubectl get clusterrole cluster-admin -o yaml
```

> **Note** After a RAM user is assigned the cluster-admin role, the RAM user has the same permissions as the Alibaba Cloud account to which the RAM user belongs. The RAM user has full control over all resources within the cluster. Proceed with caution.

```
kubectl get clusterrole cluster-admin  -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 annotations:
  rbac.authorization.kubernetes.io/autoupdate: "true"
 creationTimestamp: 2018-10-12T08:31:15Z
 labels:
  kubernetes.io/bootstrapping: rbac-defaults
 name: cluster-admin
 resourceVersion: "57"
 selfLink: /apis/rbac.authorization.k8s.io/v1/clusterroles/cluster-admin
 uid: 2f29f9c5-cdf9-11e8-84bf-00163e0b2f97
rules:
- apiGroups:
 - '*'
 resources:
 - '*'
 verbs:
 - '*'
- nonResourceURLs:
 - '*'
 verbs:
 - '*'
```

2.3. ACK default roles

When you activate Container Service for Kubernetes (ACK), you must grant default roles to a service account. Then, the service account can be used to call services, such as Elastic Compute Service (ECS), Object Storage Service (OSS), Apsara File Storage NAS, and Server Load Balancer (SLB), create clusters, and save cluster logs. This topic describes the permissions of the ACK default roles.

Role permissions

This topic describes the permissions of the following roles:

- AliyunCSManagedLogRole
- AliyunCSManagedCmsRole
- AliyunCSManagedCsiRole
- AliyunCSManagedVKRole
- AliyunCSClusterRole
- AliyunCSServerlessKubernetesRole
- AliyunCSKubernetesAuditRole
- AliyunCSManagedNetworkRole
- AliyunCSDefaultRole
- AliyunCSManagedKubernetesRole
- AliyunCSManagedArmsRole

AliyunCSManagedLogRole

The Logtail component of ACK uses AliyunCSManagedLogRole to access resources of other cloud services.

Permission	Description
log:CreateProject	Creates a project.
log:GetProject	Queries a project by name.
log:DeleteProject	Deletes a specified project.
log:CreateLogStore	Creates a Logstore in a project.
log:GetLogStore	Queries the attributes of a Logstore.
log:UpdateLogStore	Updates the attributes of a Logstore.
log:DeleteLogStore	Deletes a Logstore.
log:CreateConfig	Creates a log collection configuration.
log:UpdateConfig	Updates a log collection configuration.
log:GetConfig	Queries the details of a log collection configuration.
log:DeleteConfig	Deletes a specified log collection configuration.
log:CreateMachineGroup	Creates a machine group to apply log collection configurations.
log:UpdateMachineGroup	Updates the information of a machine group.
log:GetMachineGroup	Queries the information of a specified machine group.
log:DeleteMachineGroup	Deletes a machine group.
log:ApplyConfigToGroup	Applies a log collection configuration to a machine group.
log:GetAppliedMachineGroups	Queries the machines to which a log collection configuration is applied in a machine group.
log:GetAppliedConfigs	Queries the configurations that are applied to a machine group.
log:RemoveConfigFromMachineGroup	Removes a configuration from a machine group.
log:CreateIndex	Creates an index for a specified Logstore.
log:GetIndex	Queries the index of a specified Logstore.
log:UpdateIndex	Updates the index of a specified Logstore.
log:DeleteIndex	Deletes the index of a specified Logstore.
log:CreateSavedSearch	Creates a saved search.

Permission	Description
log:GetSavedSearch	Queries a specified saved search.
log:UpdateSavedSearch	Updates a saved search.
log:DeleteSavedSearch	Deletes a saved search.
log:CreateDashboard	Creates a dashboard.
log:GetDashboard	Queries a specified dashboard.
log:UpdateDashboard	Updates a dashboard.
log:DeleteDashboard	Deletes a dashboard.
log:CreateJob	Creates a job. For example, create an alert or a subscription.
log:GetJob	Queries a job.
log:DeleteJob	Deletes a job.
log:UpdateJob	Updates a job.
log:PostLogStoreLogs	Adds logs to a specified Logstore.
log:CreateSortedSubStore	Creates a sorted sub-Logstore.
log:GetSortedSubStore	Queries a sorted sub-Logstore.
log:ListSortedSubStore	Lists sorted sub-Logstores.
log:UpdateSortedSubStore	Updates a sorted sub-Logstore.
log:DeleteSortedSubStore	Deletes a sorted sub-Logstore.
log:CreateApp	Creates applications, such as Cost Manager and Log Audit Service.
log:UpdateApp	Updates applications, such as Cost Manager and Log Audit Service.
log:GetApp	Queries applications, such as Cost Manager and Log Audit Service.
log:DeleteApp	Deletes applications, such as Cost Manager and Log Audit Service.
cs:DescribeTemplates	Queries container templates.
cs:DescribeTemplateAttribute	Queries the attributes of a container template.

AliyunCSManagedCmsRole

The Cloud Monitor component of an ACK cluster uses AliyunCSManagedCmsRole to access resources of other cloud services.

Permission	Description
cms:DescribeMonitorGroups	Queries application groups.
cms:DescribeMonitorGroupInstances	Queries the resources in a specified application group.
cms:CreateMonitorGroup	Creates an application group.
cms:DeleteMonitorGroup	Deletes a specified application group.
cms:ModifyMonitorGroupInstances	Modifies the resources in an application group.
cms:CreateMonitorGroupInstances	Adds resources to an application group.
cms:DeleteMonitorGroupInstances	Deletes resources from an application group.
cms:TaskConfigCreate	Creates a monitoring job configuration.
cms:TaskConfigList	Lists monitoring job configurations.
cms:DescribeMetricList	Queries the time series metrics of a cloud service in a specified period.
cs:DescribeMonitorToken	Queries the token that is required to use the Cloud Monitor component.
ahas:GetSentinelAppSumMetric	Queries the metrics that are monitored by the AHAS Sentinel application.
log:GetLogStoreLogs	Queries logs in a Logstore.
slb:DescribeMetricList	Queries the time series metrics of a cloud service in a specified period.
sls:GetLogs	Queries logs in a Logstore of a specified project in Log Service.
sls:PutLogs	Updates logs in a Logstore of a specified project in Log Service.

AliyunCSManagedCsiRole

The volume plug-in of an ACK cluster uses AliyunCSManagedCsiRole to access resources of other cloud services.

- Permissions on ECS resources

Permission	Description
ecs:AttachDisk	Attaches a pay-as-you-go data disk or a system disk to an ECS instance.
ecs:DetachDisk	Detaches a pay-as-you-go disk from an ECS instance.
ecs:DescribeDisks	Queries one or more disks that you have created and local disks.

Permission	Description
ecs:CreateDisk	Creates a pay-as-you-go or subscription data disk.
ecs:ResizeDisk	Expands a disk. You can expand a system disk or a data disk.
ecs:CreateSnapshot	Creates a snapshot for a disk.
ecs:DeleteSnapshot	Deletes a specified snapshot. If you call this operation to delete a snapshot that is being created, the snapshot creation task is canceled.
ecs:CreateAutoSnapshotPolicy	Creates an automatic snapshot policy.
ecs:ApplyAutoSnapshotPolicy	Attaches an automatic snapshot policy to one or more disks.
ecs:CancelAutoSnapshotPolicy	Detaches an automatic snapshot policy from one or more disks.
ecs:DeleteAutoSnapshotPolicy	Deletes an automatic snapshot policy.
ecs:DescribeAutoSnapshotPolicyEX	Queries automatic snapshot policies that you have created.
ecs:ModifyAutoSnapshotPolicyEx	Modifies an automatic snapshot policy.
ecs:AddTags	Attaches tags to an ECS instance.
ecs:DescribeTags	Queries tags of an ECS instance.
ecs:DescribeSnapshots	Queries all the snapshots of an ECS instance or a disk.
ecs:ListTagResources	Queries tags that are attached to one or more ECS instances.
ecs:TagResources	Creates tags and attaches the tags to a specified group of ECS instances.
ecs:UntagResources	Detaches tags from a specified group of ECS instances and deletes the tags.
ecs:ModifyDiskSpec	Upgrades the performance level of an enhanced SSD.
ecs:CreateSnapshot	Creates a snapshot for a disk.
ecs:DeleteDisk	Releases a pay-as-you-go data disk.
ecs:DescribeInstanceAttribute	Queries all attributes of an ECS instance.
ecs:DescribeInstances	Queries the information of one or more ECS instances.

- Permissions on NAS file systems

Permission	Description
nas:DescribeFileSystems	Queries the descriptions of file systems.
nas:DescribeMountTargets	Queries the descriptions of mount targets.
nas:AddTags	Attaches one or more tags to a file system or overwrites the tags.
nas:DescribeTags	Queries existing tags.
nas:RemoveTags	Detaches one or more tags from a file system.
nas:CreateFileSystem	Creates a file system.
nas:DeleteFileSystem	Deletes a file system.
nas:DescribeFileSystems	Queries the descriptions of file systems.
nas:ModifyFileSystem	Modifies the information of file systems.
nas:CreateMountTarget	Creates a mount target.
nas:DeleteMountTarget	Deletes a mount target.
nas:DescribeMountTargets	Queries the descriptions of mount targets.
nas:ModifyMountTarget	Modifies the information of mount targets.

AliyunCSManagedVKRole

The Virtual Kubelet component of an ACK cluster uses AliyunCSManagedVKRole to access resources of other cloud services.

- Permissions on VPC resources

Permission	Description
vpc:DescribeVSwitches	Queries existing VSwitches.
vpc:DescribeVpcs	Queries existing VPCs.
vpc:AssociateEipAddress	Binds an elastic IP address (EIP) to a cloud service instance in the same region.
vpc:DescribeEipAddresses	Queries the EIPs that you create in a specified region.
vpc:AllocateEipAddress	Applies for an EIP.
vpc:ReleaseEipAddress	Releases a specified EIP.

- Permissions on ECS resources

Permission	Description
ecs:DescribeSecurityGroups	Queries the basic information of the security groups that you create.

Permission	Description
ecs:CreateNetworkInterface	Creates an elastic network interface (ENI).
ecs:CreateNetworkInterfacePermission	Grants permissions to create an ENI.
ecs:DescribeNetworkInterfaces	Queries ENIs.
ecs:AttachNetworkInterface	Attaches an ENI to a VPC-connected ECS instance.
ecs:DetachNetworkInterface	Detaches an ENI from an ECS instance.
ecs:DeleteNetworkInterface	Deletes an ENI.
ecs:DeleteNetworkInterfacePermission	Grants permissions to delete an ENI.

- Permissions on Alibaba Cloud DNS PrivateZone resources

Permission	Description
pvtz:AddZone	Creates a private zone.
pvtz:DeleteZone	Deletes a private zone.
pvtz:DescribeZones	Queries private zones.
pvtz:DescribeZoneInfo	Queries the information of a specified private zone.
pvtz:BindZoneVpc	Binds a private zone to a VPC or unbinds a private zone from a VPC.
pvtz:AddZoneRecord	Adds a DNS record to a private zone.
pvtz:DeleteZoneRecord	Deletes a DNS record.
pvtz:DeleteZoneRecordsByRR	Deletes DNS records.
pvtz:DescribeZoneRecordsByRR	Queries DNS records.
pvtz:DescribeZoneRecords	Queries DNS records.

- Permissions on elastic container instances (ECIs)

Permission	Description
eci:CreateContainerGroup	Creates a pod.
eci:DeleteContainerGroup	Deletes a pod.
eci:DescribeContainerGroups	Queries the information of multiple pods.
eci:DescribeContainerLog	Queries the logs of a pod.
eci:UpdateContainerGroup	Updates a pod.
eci:UpdateContainerGroupByTemplate	Updates an ECI by template.

Permission	Description
eci:CreateContainerGroupFromTemplate	Creates an ECI by template.
eci:RestartContainerGroup	Restarts an ECI.
eci:ExportContainerGroupTemplate	Exports an ECI template.
eci:DescribeContainerGroupMetric	Queries the monitoring data of an ECI.
eci:DescribeMultiContainerGroupMetric	Queries the monitoring data of multiple pods.
eci:ExecContainerCommand	Runs commands on a container.
eci:CreateImageCache	Creates an image cache.
eci:DescribeImageCaches	Queries the information of image caches.
eci:DeleteImageCache	Deletes an image cache.

AliyunCSClusterRole

When the applications are running, an ACK cluster uses AliyunCSClusterRole to access resources of other cloud services.

- Permissions on ECS resources

Permission	Description
ecs:Describe*	Queries ECS resources.

- Permissions on OSS resources

Permission	Description
oss:PutObject	Uploads a file or a folder.
oss:GetObject	Obtains a file or a folder.
oss:ListObjects	Queries files.

- Permissions on Cloud Monitor Service (CMS)

Permission	Description
cms:List*	Lists permissions on CMS resources.
cms:Get*	Obtains permissions on CMS resources.
cms:UpdateAlert	Updates an alert.
cms:CreateAlert	Creates an alert.
cms:DeleteAlert	Deletes an alert.
cms:UpdateDimensions	Updates monitoring metrics configurations.

Permission	Description
cms:CreateDimensions	Creates monitoring metrics configurations.
cms:DeleteDimensions	Deletes monitoring metrics configurations.
cms:SendAlarm	Sends a monitoring alert.
cms:CreateProject	Creates a monitoring project.
cms:DeleteProject	Deletes a monitoring project.
cms:UpdateProject	Updates a monitoring project.
cms:QueryAlarm	Queries a monitoring alert.
cms:ListAlarm	Lists monitoring alerts.
cms:CreateAlarm	Creates a monitoring alert.
cms:DeleteAlarm	Deletes a monitoring alert.
cms:UpdateAlarm	Updates a monitoring alert.

- Permissions on Server Load Balancer (SLB) resources

Permission	Description
slb:Describe*	Queries the information about an SLB instance.
slb:CreateLoadBalancer	Creates an SLB instance.
slb:DeleteLoadBalancer	Deletes an SLB instance.
slb:RemoveBackendServers	Unbinds backend servers from an SLB instance.
slb:StartLoadBalancerListener	Starts a specified listener.
slb:StopLoadBalancerListener	Stops a specified listener.
slb:CreateLoadBalancerTCPListener	Creates a TCP listener for an SLB instance.
slb:AddBackendServers*	Adds backend servers to an SLB instance.
slb:DeleteLoadBalancerListener	Deletes an SLB instance.
slb:CreateVServerGroup	Creates a VServer group and adds backend servers to the VServer group.
slb:ModifyVServerGroupBackendServers	Modifies backend servers in a VServer group.
slb:CreateLoadBalancerHTTPListener	Creates an HTTP listener for an SLB instance.
slb:SetBackendServers	Configures backend servers of an SLB instance and sets weights for the backend servers. The backend servers are ECS instances.

Permission	Description
slb:AddTags	Attaches tags to an SLB instance.

- Permissions on Log Service resources

Permission	Description
log:Get*	Obtains permissions on Log Service resources.
log:List*	Lists permissions on Log Service resources.
log:CreateProject	Creates a project.
log:DeleteProject	Deletes a specified project.
log:UpdateProject	Updates a project.
log:CreateMachineGroup	Creates a machine group to apply log collection configurations.
log:DeleteMachineGroup	Deletes a machine group.
log:UpdateMachineGroup	Updates the information of a machine group.
log:CreateLogStore	Creates a Logstore in a project.
log:DeleteLogStore	Deletes a Logstore.
log:UpdateLogStore	Updates the attributes of a Logstore.
log:CreateIndex	Creates an index for a specified Logstore.
log:DeleteIndex	Deletes the index of a specified Logstore.
log:UpdateIndex	Updates the index of a specified Logstore.
log:CreateConfig	Creates a log collection configuration.
log:DeleteConfig	Deletes a specified log collection configuration.
log:UpdateConfig	Updates a log collection configuration.
log:ApplyConfigToGroup	Applies a log collection configuration to a machine group.

AliyunCSServerlessKubernetesRole

By default, a serverless Kubernetes cluster uses AliyunCSServerlessKubernetesRole to access resources of other cloud services.

- Permissions on VPC resources

Permission	Description
DescribeVSwitches	Queries existing VSwitches.

Permission	Description
DescribeVpcs	Queries existing VPCs.
AssociateEipAddress	Binds an EIP to a cloud service instance in the same region.
DescribeEipAddresses	Queries the EIPs that you create in a specified region.
AllocateEipAddress	Applies for an EIP.
ReleaseEipAddress	Releases a specified EIP.
AddCommonBandwidthPackageIp	Binds an EIP to an EIP bandwidth plan.
RemoveCommonBandwidthPackageIp	Unbinds an EIP from an EIP bandwidth plan.

- Permissions on ECS resources

Permission	Description
DescribeSecurityGroups	Queries the basic information of the security groups that you create.
CreateNetworkInterface	Creates an ENI.
CreateNetworkInterfacePermission	Grants permissions to create an ENI.
DescribeNetworkInterfaces	Queries ENIs.
AttachNetworkInterface	Attaches an ENI to a VPC-connected ECS instance.
DetachNetworkInterface	Detaches an ENI from an ECS instance.
DeleteNetworkInterface	Deletes an ENI.
DeleteNetworkInterfacePermission	Grants permissions to delete an ENI.

- Permissions on SLB resources

Permission	Description
slb:Describe*	Queries SLB resources.
slb:CreateLoadBalancer	Creates an SLB instance.
slb:DeleteLoadBalancer	Deletes a pay-as-you-go SLB instance.
slb:RemoveBackendServers	Removes backend servers from an SLB instance.
slb:StartLoadBalancerListener	Starts a listener.
slb:StopLoadBalancerListener	Stops a listener.
slb:DeleteLoadBalancerListener	Deletes a listener of an SLB instance.

Permission	Description
slb:CreateLoadBalancerTCPListener	Creates a TCP listener for an SLB instance.
slb:AddBackendServers*	Adds backend servers to an SLB instance.
slb:UploadServerCertificate	Uploads a server certificate.
slb:CreateLoadBalancerHTTPListener	Creates an HTTP listener for an SLB instance.
slb:CreateLoadBalancerHTTPSListener	Creates an HTTPS listener for an SLB instance.
slb:CreateLoadBalancerUDPListener	Creates a UDP listener.
slb:ModifyLoadBalancerInternetSpec	Changes the billing method of a public-facing SLB instance.
slb:CreateRules	Adds forwarding rules to a specified HTTP or HTTPS listener.
slb:DeleteRules	Deletes forwarding rules.
slb:SetRule	Modifies a forwarding rule of a VServer group.
slb:CreateVServerGroup	Adds backend servers to a VServer group.
slb:SetVServerGroupAttribute	Modifies the configurations of a VServer group.
slb:AddVServerGroupBackendServers	Adds backend servers to a VServer group.
slb:RemoveVServerGroupBackendServers	Removes backend servers from a specified VServer group.
slb:ModifyVServerGroupBackendServers	Changes the backend servers of a VServer group.
slb:DeleteVServerGroup	Deletes a VServer group.
slb:SetLoadBalancerTCPListenerAttribute	Modifies the configuration of a TCP listener.
slb:SetLoadBalancerHTTPListenerAttribute	Modifies the configuration of an HTTP listener.
slb:SetLoadBalancerHTTPSListenerAttribute	Modifies the configuration of an HTTPS listener.
slb:AddTags	Adds tags to a specified SLB instance.

- Permissions on Alibaba Cloud DNS PrivateZone

Permission	Description
AddZone	Creates a private zone.
DeleteZone	Deletes a private zone.
DescribeZones	Queries private zones.
DescribeZoneInfo	Queries the information of a specified private zone.

Permission	Description
BindZoneVpc	Binds a private zone to a VPC or unbinds a private zone from a VPC.
AddZoneRecord	Adds a DNS record to a private zone.
DeleteZoneRecord	Deletes a DNS record.
DeleteZoneRecordsByRR	Deletes DNS records.
DescribeZoneRecordsByRR	Queries DNS records.
DescribeZoneRecords	Queries DNS records.

- Permissions on Container Registry (ACR) resources

Permission	Description
Get*	Queries ACR resources.
List*	Queries image repositories.
PullRepository	Pulls an image.

- Permissions on ECIs

Permission	Description
CreateContainerGroup	Creates a pod.
DeleteContainerGroup	Deletes a pod.
DescribeContainerGroups	Queries the information of multiple pods.
DescribeContainerLog	Queries the logs of a pod.
UpdateContainerGroup	Updates a pod.
UpdateContainerGroupByTemplate	Updates an ECI by template.
CreateContainerGroupFromTemplate	Creates an ECI by template.
RestartContainerGroup	Restarts an ECI.
ExportContainerGroupTemplate	Exports an ECI template.
DescribeContainerGroupMetric	Queries the monitoring data of an ECI.
DescribeMultiContainerGroupMetric	Queries the monitoring data of multiple pods.
ExecContainerCommand	Runs commands on a container.
CreateImageCache	Creates an image cache.
DescribeImageCaches	Queries the information of image caches.

Permission	Description
DeleteImageCache	Deletes an image cache.

- Permissions on RAM resources

Permission	Description
ram:PassRole	Visits the Alibaba Cloud CodePipeline console.

- Permissions on OSS resources

Permission	Description
oss:GetObject	Obtains a file or a folder.
oss:GetObjectMeta	Queries the metadata information of an object.

- Permissions on Function Compute

Permission	Description
fc:CreateService	Creates a service.
fc:ListServices	Queries services.
fc:GetService	Queries a specified service.
fc:UpdateService	Updates a specified service.
fc:DeleteService	Deletes a specified service.
fc:CreateFunction	Creates a function.
fc:ListFunctions	Queries the functions of a service.
fc:GetFunction	Queries the configurations of a specified function.
fc:GetFunctionCode	Queries the code of a function.
fc:UpdateFunction	Updates a function, including its configurations and code.
fc:DeleteFunction	Deletes a specified function.
fc:CreateTrigger	Creates a function trigger.
fc:ListTriggers	Queries the triggers of a function.
fc:GetTrigger	Queries a specified trigger.
fc:UpdateTrigger	Updates the configurations of a specified trigger.
fc:DeleteTrigger	Deletes the triggers of a specified function.
fc:PublishServiceVersion	Releases a Function Compute version.

Permission	Description
fc:ListServiceVersions	Lists Function Compute versions.
fc:DeleteServiceVersion	Deletes a Function Compute version.
fc:CreateAlias	Creates an alias and binds it to a customer master key (CMK).
fc:ListAliases	Lists all aliases of the current Alibaba Cloud account in the current region.
fc:GetAlias	Queries the information about an alias.
fc:UpdateAlias	Binds an alias to a different CMK.
fc:DeleteAlias	Deletes an alias.

AliyunCSKubernetesAuditRole

The auditing feature of ACK uses AliyunCSKubernetesAuditRole to access resources of other cloud services.

Permission	Description
log:CreateProject	Creates a project.
log:GetProject	Queries a project by name.
log:DeleteProject	Deletes a specified project.
log:CreateLogStore	Creates a Logstore in a project.
log:GetLogStore	Queries the attributes of a Logstore.
log:UpdateLogStore	Updates the attributes of a Logstore.
log:DeleteLogStore	Deletes a Logstore.
log:CreateConfig	Creates a log collection configuration.
log:UpdateConfig	Updates a log collection configuration.
log:GetConfig	Queries the details of a log collection configuration.
log:DeleteConfig	Deletes a specified log collection configuration.
log:CreateMachineGroup	Creates a machine group to apply log collection configurations.
log:UpdateMachineGroup	Updates the information of a machine group.
log:GetMachineGroup	Queries the information of a specified machine group.
log:DeleteMachineGroup	Deletes a machine group.

Permission	Description
log:ApplyConfigToGroup	Applies a log collection configuration to a machine group.
log:GetAppliedMachineGroups	Queries the machines to which a log collection configuration is applied in a machine group.
log:GetAppliedConfigs	Queries the configurations that are applied to a machine group.
log:RemoveConfigFromMachineGroup	Removes a configuration from a machine group.
log:CreateIndex	Creates an index for a specified Logstore.
log:GetIndex	Queries the index of a specified Logstore.
log:UpdateIndex	Updates the index of a specified Logstore.
log:DeleteIndex	Deletes the index of a specified Logstore.
log:CreateSavedSearch	Creates a saved search.
log:GetSavedSearch	Queries a specified saved search.
log:UpdateSavedSearch	Updates a saved search.
log:DeleteSavedSearch	Deletes a saved search.
log:CreateDashboard	Creates a dashboard.
log:GetDashboard	Queries a specified dashboard.
log:UpdateDashboard	Updates a dashboard.
log:DeleteDashboard	Deletes a dashboard.
log:CreateJob	Creates a job. For example, create an alert or a subscription.
log:GetJob	Queries a job.
log:DeleteJob	Deletes a job.
log:UpdateJob	Updates a job.
log:PostLogStoreLogs	Adds logs to a specified Logstore.

AliyunCSManagedNetworkRole

The network component of an ACK cluster uses AliyunCSManagedNetworkRole to access resources of other cloud services.

Permission	Description
ecs:CreateNetworkInterface	Creates an ENI.

Permission	Description
ecs:DescribeNetworkInterfaces	Queries ENIs.
ecs:AttachNetworkInterface	Attaches an ENI to a VPC-connected ECS instance.
ecs:DetachNetworkInterface	Detaches an ENI from an ECS instance.
ecs:DeleteNetworkInterface	Deletes an ENI.
ecs:DescribeInstanceAttribute	Queries the information of one or more ECS instances.
ecs:AssignPrivateIpAddresses	Assigns one or more secondary private IP addresses to an ENI.
ecs:UnassignPrivateIpAddresses	Unbinds one or more secondary private IP addresses from an ENI.
ecs:DescribeInstances	Queries the information of one or more ECS instances.
vpc:DescribeVSwitches	Queries the information of one or more VSwitches.

AliyunCSDefaultRole

By default, AliyunCSDefaultRole is used to access resources of other cloud services when you perform operations on ACK clusters.

- Permissions on ECS resources

Permission	Description
ecs:RunInstances	Starts an ECS instance.
ecs:RenewInstance	Renews an ECS instance.
ecs:Create*	Creates ECS resources, such as ECS instances and disks.
ecs:AllocatePublicIpAddress	Assigns a public IP address to an ECS instance.
ecs:AllocateEipAddress	Assigns an EIP to an ECS instance.
ecs:Delete*	Deletes an ECS instance.
ecs:StartInstance	Starts ECS resources.
ecs:StopInstance	Stops an ECS instance.
ecs:RebootInstance	Restarts an ECS instance.
ecs:Describe*	Queries ECS resources.
ecs:AuthorizeSecurityGroup	Sets inbound rules for a security group.
ecs:RevokeSecurityGroup	Revokes security group rules.
ecs:AuthorizeSecurityGroupEgress	Sets outbound rules for a security group.

Permission	Description
ecs:AttachDisk	Attaches a disk to an ECS instance.
ecs:DetachDisk	Detaches a disk from an ECS instance.
ecs:WaitFor*	Waits for the execution of a task.
ecs:AddTags	Adds tags to an ECS instance.
ecs:ReplaceSystemDisk	Replaces the system disk of an ECS instance.
ecs:ModifyInstanceAttribute	Modifies the attributes of an ECS instance.
ecs:JoinSecurityGroup	Adds an ECS instance to a security group.
ecs:LeaveSecurityGroup	Removes an ECS instance from a security group.
ecs:UnassociateEipAddress	Detaches an EIP from an ECS instance.
ecs:ReleaseEipAddress	Releases an EIP.
ecs:CreateKeyPair	Creates an SSH key pair.
ecs:ImportKeyPair	Imports the public key of an RSA key pair that is created by using a third-party tool.
ecs:AttachKeyPair	Attaches an SSH key pair to one or more Linux-based ECS instances.
ecs:DetachKeyPair	Detaches an SSH key pair from one or more Linux-based ECS instances.
ecs:DeleteKeyPairs	Deletes one or more SSH key pairs.
ecs:AttachInstanceRamRole	Attaches a RAM role to one or more ECS instances.
ecs:DetachInstanceRamRole	Detaches a RAM role from one or more ECS instances.
ecs:AllocateDedicatedHosts	Creates one or more pay-as-you-go or subscription dedicated hosts.
ecs:CreateOrder	Creates an order to purchase ECS instances.
ecs:DeleteInstance	Releases a pay-as-you-go instance or an expired subscription instance.
ecs:CreateDisk	Creates a pay-as-you-go or subscription data disk.
ecs:Createvpc	Creates a VPC for an ECS instance.
ecs:Deletevpc	Deletes the VPC that is connected to an ECS instance.
ecs:DeleteVSwitch	Deletes the VSwitch that is connected to an ECS instance.

Permission	Description
ecs:ResetDisk	Rolls back a disk to a specified state by using a disk snapshot.
ecs:DeleteSnapshot	Deletes a specified snapshot.
ecs:AllocatePublicIpAddress	Assigns a public IP address to an ECS instance.
ecs:CreateVSwitch	Creates a VSwitch for an ECS instance.
ecs:DeleteSecurityGroup	Deletes a security group.
ecs:CreateImage	Creates a custom image.
ecs:RemoveTags	Deletes tags from an ECS instance.
ecs:ReleaseDedicatedHost	Releases a pay-as-you-go dedicated host.
ecs:CreateInstance	Creates a subscription or pay-as-you-go ECS instance.
ecs:RevokeSecurityGroupEgress	Deletes an outbound rule of a security group. This revokes outbound permissions of the security group.
ecs:DeleteDisk	Releases a pay-as-you-go data disk.
ecs:StopInstance	Stops an ECS instance.
ecs:CreateSecurityGroup	Creates a security group.
ecs:RevokeSecurityGroup	Deletes an inbound rule of a security group. This revokes inbound permissions of the security group.
ecs:DeleteImage	Deletes a custom image.
ecs:ModifyInstanceSpec	Modifies the instance type of ECS instances or public bandwidth of a pay-as-you-go ECS instance.
ecs:CreateSnapshot	Creates a snapshot for a disk.
ecs:CreateCommand	Creates a Cloud Assistant command.
ecs:InvokeCommand	Triggers a Cloud Assistant command on one or more ECS instances.
ecs:StopInvocation	Stops the process of a running Cloud Assistant command on one or more ECS instances.
ecs:DeleteCommand	Deletes a Cloud Assistant command.
ecs:RunCommand	Creates a Cloud Assistant command of the shell, PowerShell, or batch type, and runs the command on one or more ECS instances.
ecs:DescribeInvocationResults	Queries the result of running a Cloud Assistant command on a specified ECS instance.

Permission	Description
ecs:ModifyCommand	Modifies a Cloud Assistant command.

- Permissions on VPC resources

Permission	Description
vpc:Describe*	Queries VPC resources.
vpc:AllocateEipAddress	Assigns an EIP to an ECS instance.
vpc:AssociateEipAddress	Binds an EIP to an ECS instance.
vpc:UnassociateEipAddress	Unbinds an EIP from an ECS instance.
vpc:ReleaseEipAddress	Releases an EIP.
vpc:CreateRouteEntry	Creates a route entry.
vpc:DeleteRouteEntry	Deletes a route entry.
vpc:CreateVSwitch	Creates a VSwitch.
vpc:DeleteVSwitch	Deletes a VSwitch.
vpc:CreateVpc	Creates a VPC.
vpc:DeleteVpc	Deletes a VPC.
vpc:CreateNatGateway	Creates a network address translation (NAT) gateway.
vpc:DeleteNatGateway	Deletes a specified NAT gateway.
vpc:CreateSnatEntry	Adds a source network address translation (SNAT) entry to a specified SNAT table.
vpc:DeleteSnatEntry	Deletes an SNAT entry from a specified SNAT table.
vpc:ModifyEipAddressAttribute	Modifies the name, description, and peak bandwidth of a specified EIP.
vpc:CreateForwardEntry	Adds a destination network address translation (DNAT) entry to a specified DNAT table.
vpc:DeleteBandwidthPackage	Creates a NAT service plan.
vpc:CreateBandwidthPackage	Deletes a NAT service plan.
vpc:DeleteForwardEntry	Deletes a DNAT entry from a specified DNAT table.
vpc:TagResources	Creates tags and attaches them to a specified resource.

Permission	Description
vpc:DeletionProtection	Enables or disables deletion protection for an instance.

- Permissions on SLB resources

Permission	Description
slb:Describe*	Queries the information about an SLB instance.
slb:CreateLoadBalancer	Creates an SLB instance.
slb:DeleteLoadBalancer	Deletes an SLB instance.
slb:RemoveBackendServers	Unbinds backend servers from an SLB instance.
slb:StartLoadBalancerListener	Starts a specified listener.
slb:StopLoadBalancerListener	Stops a specified listener.
slb:CreateLoadBalancerTCPListener	Creates a TCP listener for an SLB instance.
slb:AddBackendServers	Adds backend servers to an SLB instance.
slb:CreateVServerGroup	Creates a VServer group and adds backend servers to the VServer group.
slb:CreateLoadBalancerHTTPSListener	Creates an HTTPS listener for an SLB instance.
slb:CreateLoadBalancerUDPListener	Creates a UDP listener.
slb:ModifyLoadBalancerInternetSpec	Changes the billing method of a public-facing SLB instance.
slb:SetBackendServers	Configures backend servers of an SLB instance and sets weights for the backend servers. The backend servers are ECS instances.
slb:AddVServerGroupBackendServers	Adds backend servers to a VServer group.
slb:DeleteVServerGroup	Deletes a VServer group.
slb:ModifyVServerGroupBackendServers	Changes the backend servers of a VServer group.
slb:CreateLoadBalancerHTTPListener	Creates an HTTP listener for an SLB instance.
slb:RemoveVServerGroupBackendServers	Removes backend servers from a specified VServer group.
slb:DeleteLoadBalancerListener	Deletes a listener of an SLB instance.
slb:AddTags	Adds tags to a specified SLB instance.
slb:RemoveTags	Removes tags from a specified SLB instance.

Permission	Description
slb:SetLoadBalancerDeleteProtection	Enables or disables deletion protection for an SLB instance.

- Permissions on Domain Name System (DNS) resources

Permission	Description
dns:Describe*	Queries DNS resources.
dns:AddDomainRecord	Adds a DNS record.

- Permissions on RDS resources

Permission	Description
rds:Describe*	Queries RDS resources.
rds:ModifySecurityIps	Modifies the IP address whitelist of an RDS instance.

- Permissions on Resource Orchestration Service (ROS)

Permission	Description
ros:Describe*	Queries ROS resources.
ros:WaitConditions	Waits for the execution of an ROS script.
ros:AbandonStack	Stops a stack.
ros:DeleteStack	Deletes a stack.
ros:CreateStack	Creates a stack.
ros:UpdateStack	Updates a stack.
ros:ValidateTemplate	Validates an ROS template.
ros:DoActions	Performs actions.
ros:InquiryStack	Queries a stack.
ros:SetDeletionProtection	Enables or disables deletion protection.
ros:PreviewStack	Previews a stack.

- Permissions on Auto Scaling (ESS)

Permission	Description
ess:Describe*	Queries ESS resources.
ess:CreateScalingConfiguration	Creates a scaling configuration.

Permission	Description
ess:EnableScalingGroup	Enables a scaling group.
ess:ExitStandby	Switches the status of a standby ECS instance in a scaling group to running.
ess:DetachDBInstances	Removes one or more RDS instances from a scaling group.
ess:DetachLoadBalancers	Removes one or more SLB instances from a scaling group.
ess:AttachInstances	Adds one or more ECS instances to a scaling group.
ess:DeleteScalingConfiguration	Deletes a scaling configuration.
ess:AttachLoadBalancers	Adds one or more SLB instances.
ess:DetachInstances	Removes one or more ECS instances from a scaling group.
ess:ModifyScalingRule	Modifies a scaling group rule.
ess:RemoveInstances	Removes ECS instances from a specified scaling group.
ess:ModifyScalingGroup	Modifies a scaling group.
ess:AttachDBInstances	Adds one or more RDS instances.
ess:CreateScalingRule	Creates a scaling rule.
ess:DeleteScalingRule	Deletes a scaling rule.
ess:ExecuteScalingRule	Runs a scaling rule.
ess:SetInstancesProtection	Enables or disables protection for one or more ECS instances in a scaling group.
ess:ModifyNotificationConfiguration	Modifies a notification configuration for auto scaling events and resource changes.
ess:CreateNotificationConfiguration	Creates a notification configuration for auto scaling events and resource changes.
ess:EnterStandby	Switches the status of an ECS instance in the scaling group to standby.
ess:DeleteScalingGroup	Deletes a scaling group.
ess:CreateScalingGroup	Creates a scaling group.
ess:DeleteNotificationConfiguration	Deletes a notification configuration for auto scaling events and resource changes.
ess:DisableScalingGroup	Disables a scaling group.

Permission	Description
ModifyScalingConfiguration	Modifies a scaling configuration.
SetGroupDeletionProtection	Enables or disables deletion protection for a scaling group.

- Permissions on RAM resources

Permission	Description
ram:PassRole	Authorizes a RAM user to use other cloud services.
ram:Get*	Queries permissions on RAM resources.
ram:List*	Lists permissions on RAM resources.
ram:DetachPolicyFromRole	Revokes a specified permission from a role.
ram:AttachPolicyToRole	Grants a permission to a specified role.
ram:DeletePolicy	Deletes a specified permission policy.
ram:DeletePolicyVersion	Deletes a policy of a specified version.
ram:DeleteRole	Deletes a RAM role.
ram:CreateRole	Creates a RAM role.
ram:CreatePolicy	Creates a RAM policy.
ram:CreateServiceLinkedRole	Creates permissions to be granted to service linked roles.

- Permissions on CMS resources

Permission	Description
cms:CreateMyGroups	Creates private application groups.
cms:AddMyGroupInstances	Adds resources to a private application group.
cms:DeleteMyGroupInstances	Deletes resources from a private application group.
cms:DeleteMyGroups	Deletes private application groups.
cms:GetMyGroups	Queries private application groups.
cms:ListMyGroups	Lists private application groups.
cms:UpdateMyGroupInstances	Updates resources in a private application group.
cms:UpdateMyGroups	Updates private application groups.
cms:TaskConfigCreate	Creates a monitoring job configuration.

Permission	Description
cms:TaskConfigList	Lists monitoring job configurations.

- Permissions on ESS resources

Permission	Description
ess:CreateLifecycleHook	Creates one or more lifecycle hooks for a scaling group.
ess:DescribeLifecycleHooks	Queries lifecycle hooks.
ess:ModifyLifecycleHook	Modifies a lifecycle hook.
ess:DeleteLifecycleHook	Deletes a lifecycle hook.

- Permissions on Edge Node Service (ENS) resources

Permission	Description
ens:Describe*	Queries the permissions on ENS resources.
ens:CreateInstance	Creates an ENS instance.
ens:StartInstance	Starts an ENS instance.
ens:StopInstance	Stops an ENS instance.
ens:ReleasePrePaidInstance	Releases a subscription instance.

AliyunCSManagedKubernetesRole

A managed Kubernetes cluster uses AliyunCSManagedKubernetesRole to access resources of other cloud services.

- Permissions on ECS resources

Permission	Description
ecs:Describe*	Queries ECS resources.
ecs:CreateRouteEntry	Creates a route entry.
ecs:DeleteRouteEntry	Deletes a route entry.
ecs:CreateNetworkInterface	Creates an ENI.
ecs:DeleteNetworkInterface	Deletes an ENI.
ecs:CreateNetworkInterfacePermission	Grants permissions to create an ENI.
ecs:DeleteNetworkInterfacePermission	Grants permissions to delete an ENI.
ecs:ModifyInstanceAttribute	Modifies the attributes of an instance.

Permission	Description
ecs:AttachKeyPair	Attaches an SSH key pair to one or more Linux-based ECS instances.
ecs:StopInstance	Stops an instance.
ecs:StartInstance	Starts an instance.
ecs:ReplaceSystemDisk	Replaces the system disk or the operating system of an ECS instance.

- Permissions on SLB resources

Permission	Description
slb:Describe*	Queries SLB resources.
slb:CreateLoadBalancer	Creates an SLB instance.
slb:DeleteLoadBalancer	Deletes an SLB instance.
slb:ModifyLoadBalancerInternetSpec	Changes the billing method of a public-facing SLB instance.
slb:RemoveBackendServers	Removes backend servers from an SLB instance.
slb:AddBackendServers	Adds backend servers to an SLB instance.
slb:RemoveTags	Removes tags from a specified SLB instance.
slb:AddTags	Adds tags to a specified SLB instance.
slb:StopLoadBalancerListener	Stops a listener.
slb:StartLoadBalancerListener	Starts a listener.
slb:SetLoadBalancerHTTPListenerAttribute	Modifies the configuration of an HTTP listener.
slb:SetLoadBalancerHTTPSListenerAttribute	Modifies the configuration of an HTTPS listener.
slb:SetLoadBalancerTCPListenerAttribute	Modifies the configuration of a TCP listener.
slb:SetLoadBalancerUDPListenerAttribute	Modifies the configuration of a UDP listener.
slb:CreateLoadBalancerHTTPSListener	Creates an HTTPS listener for an SLB instance.
slb:CreateLoadBalancerHTTPListener	Creates an HTTP listener for an SLB instance.
slb:CreateLoadBalancerTCPListener	Creates a TCP listener for an SLB instance.
slb:CreateLoadBalancerUDPListener	Creates a UDP listener.
slb:DeleteLoadBalancerListener	Deletes a listener of an SLB instance.
slb:CreateVServerGroup	Adds backend servers to a VServer group.

Permission	Description
slb:DescribeVServerGroups	Queries VServer groups.
slb:DeleteVServerGroup	Deletes a VServer group.
slb:SetVServerGroupAttribute	Modifies the configurations of a VServer group.
slb:DescribeVServerGroupAttribute	Queries the information about a VServer group.
slb:ModifyVServerGroupBackendServers	Changes the backend servers of a VServer group.
slb:AddVServerGroupBackendServers	Adds backend servers to a VServer group.
slb:ModifyLoadBalancerInstanceSpec	Modifies the specifications of an SLB instance.
slb:ModifyLoadBalancerInternetSpec	Changes the billing method of a public-facing SLB instance.
slb:RemoveVServerGroupBackendServers	Removes backend servers from a specified VServer group.

- Permissions on VPC resources

Permission	Description
vpc:Describe*	Queries VPC resources.
vpc:DeleteRouteEntry	Deletes a custom route entry.
vpc:CreateRouteEntry	Creates a custom route entry.

- Permissions on ACR resources

Permission	Description
cr:Get*	Queries ACR resources.
cr:List*	Queries image repositories.
cr:PullRepository	Pulls an image.

AliyunCSManagedArmsRole

The application real-time monitoring agent of an ACK cluster uses AliyunCSManagedArmsRole to access resources of other cloud services.

Permission	Description
arms:CreateApp	Creates an application monitoring job.
arms:DeleteApp	Deletes an application monitoring job.
arms:ConfigAgentLabel	Modifies the tags of the application monitoring agent.

Permission	Description
arms:GetAssumeRoleCredentials	Queries the key that is required for a RAM user to assume a RAM role during application monitoring.
arms:CreateProm	Creates a monitoring job based on Alibaba Cloud Prometheus Monitoring.
arms:SearchEvents	Queries alert events.
arms:SearchAlarmHistories	Queries the records of sending alerts.
arms:SearchAlertRules	Queries monitoring alert rules.
arms:GetAlertRules	Obtains monitoring alert rules.
arms:CreateAlertRules	Creates monitoring alert rules.
arms:UpdateAlertRules	Updates monitoring alert rules.
arms:StartAlertRule	Enables a monitoring alert rule.
arms:StopAlertRule	Disables a monitoring alert rule.
arms:CreateContact	Creates an alert contact.
arms:SearchContact	Queries an alert contact.
arms:UpdateContact	Updates an alert contact.
arms:CreateContactGroup	Creates an alert contact group.
arms:SearchContactGroup	Queries an alert contact group.
arms:UpdateContactGroup	Updates an alert contact group.

Related information

- Quick start for first-time users

2.4. The service linked role for ACK

This topic describes the service linked role AliyunServiceRoleForContainerService for Alibaba Cloud Container Service for Kubernetes (ACK) and how to delete the service linked role.

Background

AliyunServiceRoleForContainerService is a Resource Access Management (RAM) role. Alibaba Cloud Container Service for Kubernetes (ACK) assumes this service linked role to obtain the permissions required for accessing other Alibaba Cloud services when you use certain ACK features. For more information, see Service-linked roles.

Scenarios

AliyunServiceRoleForContainerService is automatically created to grant ACK the permissions required for accessing other Alibaba Cloud services, such as Server Load Balancer (SLB), Auto Scaling (ESS), Elastic Compute Service (ECS), Virtual Private Cloud (VPC), and Resource Orchestration Service (ROS).

Description

The AliyunServiceRoleForContainerService role grants ACK the permissions to access the following Alibaba Cloud services: For more information, see Content of the AliyunServiceRoleForContainerService permission policy.

ECS VPC SLB DNS RDS ROS ESS RAM CMS ENS

Delete the service-linked role for ACK

After you delete the service linked role AliyunServiceRoleForContainerService for ACK, you are no longer authorized to manage ACK clusters, for example, you cannot create, expand, or scale ACK clusters. Take the following steps to delete AliyunServiceRoleForContainerService:

> **Notice** Before you delete AliyunServiceRoleForContainerService, make sure that you have deleted all ACK clusters under the current account. The deletion failsif clusters still exist under the account.

1. Log on to the RAM console.
2. In the left-side navigation pane, click **RAM Roles**.
3. On the **RAM Roles** page, enter AliyunServiceRoleForContainerService in the search bar and click the search icon.
4. Find AliyunServiceRoleForContainerService and click **Delete** in the Actions column.
5. In the **Delete RAM Role** message, click **OK**.
   - To delete AliyunServiceRoleForContainerService under the current account, you must first delete the existing ACK clusters. Otherwise, the deletion fails.
   - If no ACK cluster exists under the current account, the deletion succeeds.

FAQ

Why is AliyunServiceRoleForContainerService not automatically created for a RAM user account?

AliyunServiceRoleForContainerService is not created for a RAM user account because the RAM user account does not have the required permission. You must attach the following permission policy to the RAM user account:

```
{
  "Statement": [
    {
      "Action": [
        "ram:CreateServiceLinkedRole"
      ],
      "Resource": "acs:ram:*:<The ID of the Alibaba Cloud account>:role/*",
      "Effect": "Allow",
      "Condition": {
        "StringEquals": {
          "ram:ServiceName": [
            "cs.aliyuncs.com"
          ]
        }
      }
    }
  ],
  "Version": "1"
}
```

> **Note** Enter the *ID of the Alibaba Cloud account* that creates the RAM user account into the content of the preceding permission policy.

Related information

- Service-linked roles

2.5. FAQ about authorization management

This topic provides answers to some frequently asked questions about authorization management.

Can I grant permissions on applications?

Yes. You can grant permissions on applications. You can create a custom ClusterRole and define a rule to grant permissions on individual applications. You can use the `resourceNames` field to specify the applications.

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Authorizations**.
3. On the **Authorizations** page, select the RAM user that you want to manage on the **Select RAM User** wizard page and click **Modify Permissions**.

   > **Note** If you log on as a RAM user, make sure that the RAM user has at least read-only permissions on the cluster that you want to manage. In addition, the RAM user must be assigned the cluster-admin role or predefined RBAC administrator role. For more information, see Create a custom RAM policy.

4. On the **Configure Role-Based Access Control (RBAC)** wizard page, click **Add Permissions**.

   Select the cluster and namespace, and then select **Custom**. Select the ClusterRole that you want to manage from the **Custom** drop-down list and click **Next Step**.

> **Note** You can assign one predefined RBAC role and one or more custom RBAC roles in the specified cluster and namespace to a RAM user.

The following table describes the permissions that the predefined and custom RBAC roles have on clusters and namespaces.
Roles and permissions

Role	RBAC permissions on cluster resources
Administrator	Read and write permissions on resources in all namespaces.
O&M Engineer	Read and write permissions on visible resources in the console in all namespaces and read-only permissions on nodes, persistent volumes (PVs), namespaces, and quotas.
Developer	Read and write permissions on visible resources in the console in all or specified namespaces.
Restricted User	Read-only permissions on visible resources in the console in all or specified namespaces.
Custom	The permissions of a custom role are determined by the ClusterRole that you select. Before you select a ClusterRole, check the permissions of the ClusterRole and make sure that you grant only the required permissions to the RAM user. For more information, see Create a custom RAM policy.

For more information about the subsequent steps, see Assign RBAC roles to a RAM user.

How do I grant a RAM user the permissions to create clusters?

1. Use your Alibaba Cloud account to grant permissions to the system roles used by Container Service for Kubernetes (ACK).
   - You need only to grant permissions to the system roles once. If you are not sure whether the permissions are granted, log on using your Alibaba Cloud account and see: https://ur.alipay.com/1paTcxSWdAEW70GVH5TZiO.
   - For more information about the default system roles, see ACK default roles.
2. Use your Alibaba Cloud account to assign custom RAM policies to the RAM user.

Make sure that the RAM user has the `cs:CreateCluster` permission. For more information, see 自定义RAM授权策略.
The following YAML template is provided as an example.

```
{
 "Statement": [{
   "Action": [
     "cs:*"
   ],
   "Effect": "Allow",
   "Resource": [
     "CreateCluster"
   ]
 }],
 "Version": "1"
}
```

> **Note**
> - When you create a cluster, you must assign cloud resources such as virtual private cloud (VPC) resources to the cluster. Make sure that the RAM user is granted the required permissions to access cloud resources.
> - Make sure that the RAM user has the List permission on VPC resources. To grant this permission, you can attach the AliyunVPCReadOnlyAccess policy to the RAM user.
> - If you want to configure other resources for the cluster, check the corresponding permission policy or documents on authorization. For more information, see RAM authorization.

How can I go to the RAM authorization page?

If you revoked the permissions granted to the system roles used by ACK, you must grant the permissions again.

For more information, see Step 2: Assign the default roles.

> **Note** You must use your Alibaba Cloud account to grant the permissions.

Why is a RAM user that has the `cs:admin` permission fail to create custom resource definitions (CRDs) in ACK?

The default administrator of clusters that are created before May 2019 does not have access permissions on all Kubernetes resources. To fix this issue, you can assign a custom `cluster-admin` role to the RAM user, or delete the existing `cs:admin` ClusterRole and create a ClusterRole.

The following YAML template is provided as an example.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: cs:admin
rules:
- apiGroups:
 - '*'
 resources:
 - '*'
 verbs:
 - '*'
- nonResourceURLs:
 - '*'
 verbs:
 - '*'
```

What do I do if the APISERVER_403 error occurs?

The current RAM user is not granted the required RBAC permissions on the Kubernetes cluster. You must go to the **Authorizations** page to grant permissions to the RAM user. For more information, see Assign RBAC roles to a RAM user.

How do I reassign a RAM role to an ECS instance?

When the application that runs on an Elastic Compute Service (ECS) instance sends requests to `metadata api 100`, the **404** error code or the following error message is returned: **Message:Node condition RAMRoleError is now: True, reason: NodeHasNoRAMRole**. You can reassign a RAM role to an ECS instance by using the following methods:

- If the RAM role of an ECS instance is deleted, you must reassign the RAM role of the corresponding node to the ECS instance. For more information, see Replace an instance RAM role.
  - If the ECS instance serves as a master node in your cluster, you must assign the master role to the ECS instance. Choose **Cluster Information > Cluster Resources > Master RAM Role**.
  - If the ECS instance serves as a worker node in your cluster, you must assign the worker role to the ECS instance. Choose **Cluster Information > Cluster Resources > Worker RAM Role**.
- If you modified the policies of the RAM role, check whether the modified content is valid.
- If you modified the policies of the RAM role before errors occurred, try to roll back the policies to the original version.

The RAM user is granted read-only permissions on all clusters but still fail to query all clusters

Problem

The RAM user is granted read-only permissions on all clusters by using the RAM console, and access permissions on specified namespaces of two clusters by using RBAC. Previously, the RAM user can query all clusters in the console. However, the RAM user can query only some of the clusters now. The permissions of the RAM user are not recently modified.

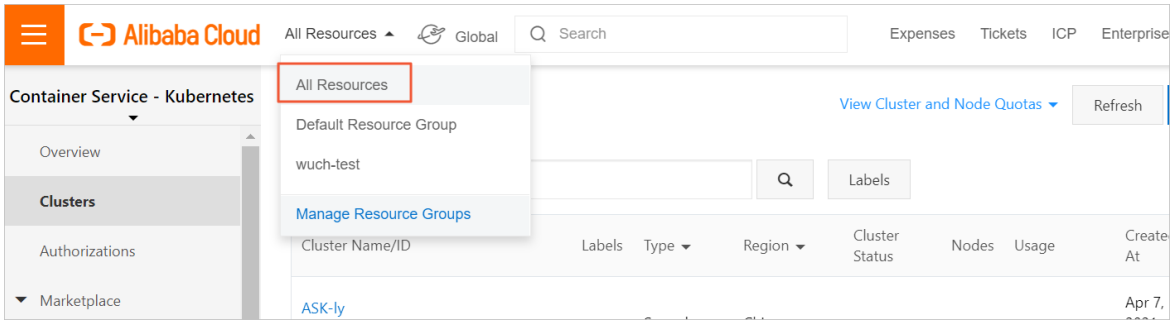
Cause

You did not log on to the console by using the RAM user that has read-only permissions on all clusters. Alternatively, you did not select to display **All Resources** in the console.

Solution

1. Log on to the ACK console.

2. In the top navigation bar, choose **All Resources > All Resources**.



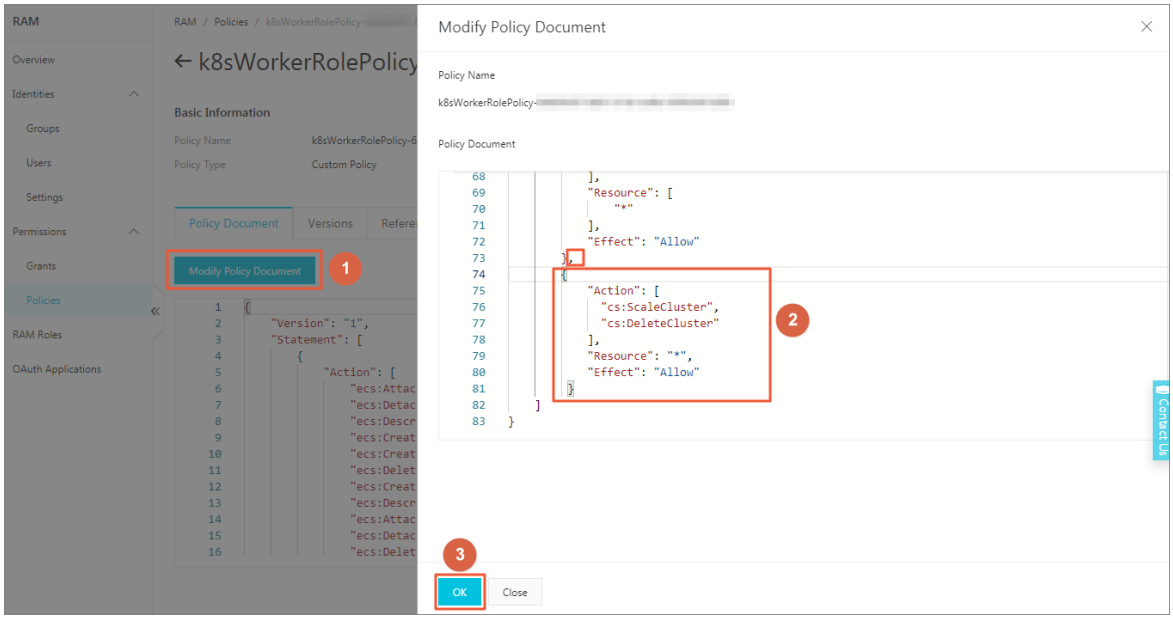
3. Move the pointer over the avatar in the upper-right corner and make sure that you are logged on as the RAM user.

How do I assign custom RAM roles in an ACK cluster?

You cannot assign custom RAM roles in an ACK cluster. However, you can attach custom permission policies to the worker RAM role that is automatically created when you create worker nodes in the cluster.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the **details** page, click **Cluster Information**.
5. On the **Cluster Information** page, click the **Cluster Resources** tab and click the link to the right of **Worker RAM Role**.
6. You are redirected to the RAM console. On the **Permissions** tab, click the name in the Policy column.
7. On the **details** page of the policy, click **Modify Policy Document**. In the Modify Policy Document pane, paste the following content into the Policy Document code editor and click **OK**. In this example, the permissions to scale and delete clusters are added to the policy. For more information, see 自定义RAM授权策略.

```
{
      "Action": [
       "cs:ScaleCluster",
       "cs:DeleteCluster"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
```



How does a RAM user assign RBAC roles to other RAM users?

By default, you can use only an Alibaba Cloud account to assign RBAC roles to other RAM users. To allow a RAM user to assign RBAC roles to other RAM users, you must first assign the predefined RBAC administrator role or cluster-admin role to the RAM user. This way, the RAM user has the permissions to manage the cluster or namespace. In addition, you must attach a RAM permission policy to the RAM user. The permission policy must contain the following content:

- The permissions to view other RAM users that belong to the current Alibaba Cloud account.
- The permissions to attach permission policies to other RAM users.
- The permissions to view configurations of RBAC roles.
- The permissions to assign RBAC roles to other RAM users.

1. Grant RAM permissions to the RAM user:

   Log on to the RAM console and grant RAM permissions to the RAM user. For more information, see 自定义RAM授权策略.
   Example:

```
{
  "Statement": [{
      "Action": [
        "ram:Get*",
        "ram:List*",
        "cs:GetUserPermissions",
        "cs:GetSubUsers",
        "cs:GrantPermission"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ram:AttachPolicyToUser",
        "ram:AttachPolicy"
      ],
      "Effect": "Allow",
      "Resource": [
        "acs:ram:*:*:policy/xxxxxx",
        "acs:*:*:*:user/*"
      ]
    }
  ],
  "Version": "1"
}
```

> **Note** Replace *xxxxxx* with the name of the permission policy that you want to allow the RAM user to attach to other RAM users. If you replace xxxxxx with an asterisk (*), it indicates that the RAM user is authorized to attach all permission policies to other RAM users.

2. Use the RAM user to grant permissions to other RAM users.

   After the RAM user is attached with the preceding policy, the RAM user is authorized to attach the specified permission policies to other RAM users and assign RBAC roles to other RAM users. For more information about how to assign RBAC roles to a RAM user, see Assign RBAC roles to a RAM user.

3.Cluster

3.1. Migrate to professional managed Kubernetes clusters

3.1.1. Hot migration from dedicated Kubernetes clusters to professional managed Kubernetes clusters

Container Service for Kubernetes (ACK) supports hot migration from dedicated Kubernetes clusters to professional managed Kubernetes clusters. You can dynamically migrate from existing dedicated Kubernetes clusters to professional managed Kubernetes clusters. This way, you can benefit from the features that are provided by professional managed Kubernetes clusters. This topic describes how to migrate from a dedicated Kubernetes cluster to a professional managed Kubernetes cluster.

Prerequisites

Before you can migrate from a dedicated Kubernetes cluster to a professional managed Kubernetes cluster, an Object Storage Service (OSS) bucket must be created. For more information about how to create an OSS bucket, see Create buckets.

> **Note** To migrate from a dedicated Kubernetes cluster to a professional managed Kubernetes cluster, you must first Submit a ticket to apply for this feature to be enabled on your account.

Precaution

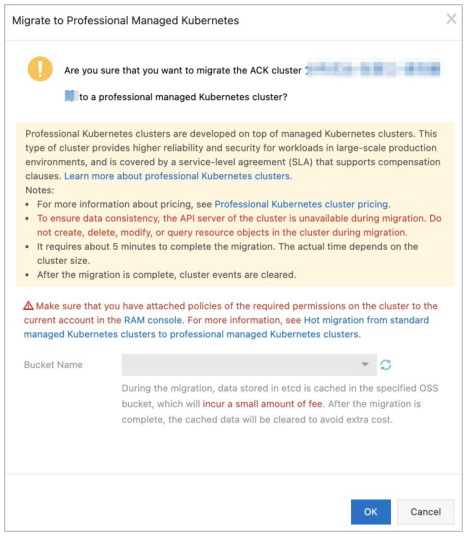
- Make sure that the Kubernetes version of your dedicated Kubernetes cluster is V1.16 or later. Otherwise, you must upgrade the Kubernetes version. For more information about how to upgrade the Kubernetes version, see Upgrade a cluster.
- Make sure that the following pods are migrated to worker nodes: pods of control plane components, such as the API server, Kube Controller Manager, the cloud controller manager, and kube-scheduler, and pods in the kube-system namespace, excluding DaemonSet pods. Control plane components are replaced by managed components after the migration.
- The master nodes no longer belong to the cluster and change to the **Unknown** state after the migration is completed.
- You cannot roll back the migration after the migration is completed.
- The Elastic Compute Service (ECS) instances in the dedicated Kubernetes cluster are not automatically deleted after the migration is completed. You must manually delete the ECS instances.

Hot migration from dedicated Kubernetes clusters to professional managed Kubernetes clusters

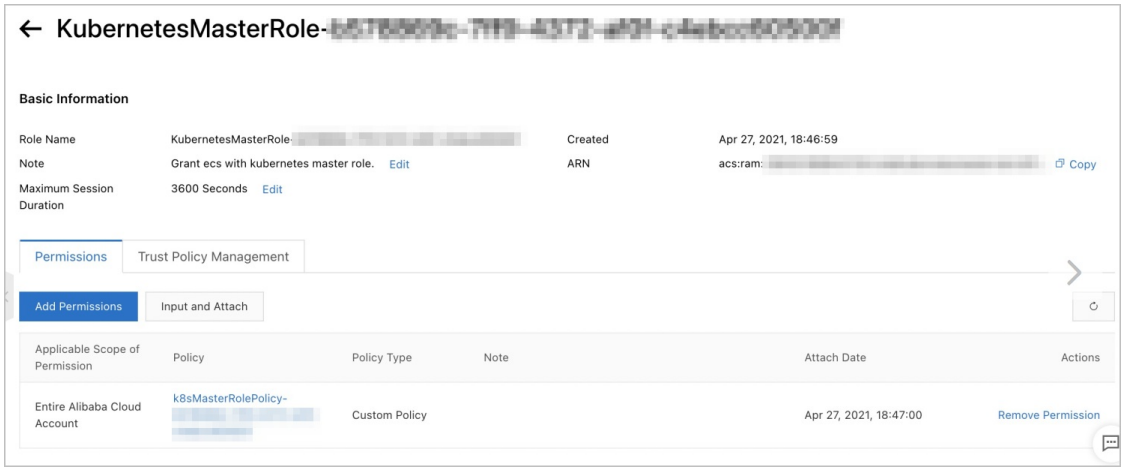
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the Clusters page, find the dedicated Kubernetes cluster that you want to migrate and click **Migrate to Professional Managed Kubernetes** in the **Actions** column.
4. In the **Migrate to Professional Managed Kubernetes** dialog box, grant the permissions that are required to perform the migration and then click **OK**.

To migrate from a dedicated Kubernetes cluster to a professional managed Kubernetes cluster, you must perform the following operations to grant the required permissions.

i. In the **Migrate to Professional Managed Kubernetes** dialog box, click **RAM console**.



ii. On the details page of the master Resource Access Management (RAM) role, click the policy whose name starts with **k8sMasterRolePolicy**.



iii. On the details page of this policy, click **Modify Policy Document** on the **Policy Document** tab.

iv. In the **Modify Policy Document** panel, add the following content to the **Statement** section and then click **OK**.

```
,
    {
      "Action": [
        "oss:PutObject",
        "oss:GetObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "acs:oss:*:*:<YOUR_BUCKET_NAME>/*" # Replace <YOUR_BUCKET_NAME> with the name of the O
SS bucket that you selected in the Migrate to Professional Managed Kubernetes dialog box.
      ]
    }
```

The following figure shows the modified policy content.

After the migration is completed, the master nodes change to the **Unknown** state. You can perform the following steps to check the node states.

i. On the Clusters page, find the cluster that you want to manage and click **Details** in the **Actions** column.

ii. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.

iii. On the Nodes page, check the states of master nodes in the **Role/Status** column.

Delete master nodes after the hot migration

After you migrate from a dedicated Kubernetes cluster to a professional managed Kubernetes cluster, you can delete the master nodes from the cluster. ACK does not allow you to delete master nodes in the console. You can run **kubectl** commands to delete master nodes.

Before you run commands, make sure that a **kubectl** client is connected to your cluster. For more information about how to use kubectl to connect to a cluster, see Connect to Kubernetes clusters by using kubectl.

1. Run the following command to query the names of master nodes that you want to delete:

```
kubectl get node | grep master
```

2. Run the following command to delete a master node:

```
kubectl delete node <MASTER_NAME>
```

> **Note** Replace <MASTER_NAME> with the name of a master node that you obtained in the previous step.

References

You can migrate from standard managed Kubernetes clusters to professional managed Kubernetes clusters. For more information, see 热迁移ACK标准托管集群至Pro托管集群.

3.2. Operating system

3.2.1. CIS reinforcement

You can enable Center for Internet Security (CIS) reinforcement to enhance the security of the operating systems of nodes in a Container Service for Kubernetes (ACK) cluster. This topic describes how ACK implements CIS reinforcement based on the Alibaba Cloud Linux 2 operating system and how to assess CIS Benchmark configuration recommendations.

Context

CIS is a third-party security organization that is committed to leading a global community of businesses, public service sectors, and academia to develop security best practice solutions. CIS provides CIS Benchmarks for the Linux-based operating systems released by major companies, such as Alibaba Cloud Linux 2, CentOS, and Ubuntu. CIS Benchmarks have become a critical criterion for assessing OS security for many Alibaba Cloud customers. For more information, see CIS WorkBench.

Alibaba Cloud Linux 2 is the official OS image developed by Alibaba Cloud and the default OS image used in ACK clusters. Alibaba Cloud Linux 2 passed the certification procedure of CIS on August 16, 2019. CIS then released CIS Aliyun Linux 2 Benchmark version 1.0.0. For more information, see CIS Aliyun Linux 2 Benchmark version 1.0.0.

CIS Aliyun Linux 2 Benchmark

The latest version of CIS Aliyun Linux 2 Benchmark is V1.0.0. To download CIS Aliyun Linux 2 Benchmark, see Download CIS Aliyun Linux 2 Benchmark version 1.0.0. CIS Aliyun Linux 2 Benchmark version 1.0.0 consists of 204 items that are classified into two security levels. Level 1 contains 168 items, and Level 2 contains 36 items. Differences between Level 1 and Level 2 items:

- Level 1 items are used to implement basic improvements. These items do not have a large impact on system performance.
- Level 2 items are suitable for scenarios that require high security. These items may increase performance overhead.

Besides, CIS Aliyun Linux 2 Benchmark classifies the items into two groups based on scoring information: Scored and Not Scored.

- Scored: Compliance with Scored items increases the final benchmark score. Failure to comply with Scored items decreases the final benchmark score.
- Not Scored: Compliance with Not Scored items does not increase the final benchmark score. Failure to comply with Not Scored items does not decrease the final benchmark score.

Therefore, the 204 items of CIS Aliyun Linux 2 Benchmark can be classified into four groups:

- Level 1 Scored (145 items)
- Level 1 Not Scored (21 items)
- Level 2 Scored (33 items)
- Level 2 Not Scored (3 items)

Level 2 items may negatively impact system performance and Not Scored items do not affect the final benchmark score. Therefore, **ACK provides reinforcement for only Level 1 Scored items**.

Enable CIS reinforcement

You can choose to enable CIS reinforcement when you create an ACK cluster. This way, the system automatically configures CIS reinforcement for the cluster. This ensures that the Alibaba Cloud Linux 2 images of all nodes in the cluster meet most requirements of Level 1 Scored items in CIS Aliyun Linux 2 Benchmark version 1.0.0. For more information about the required items, see CIS Level 1 Scored items that are covered by CIS reinforcement.

> **Note** To meet the requirements of Level 1 items, ACK automatically creates a normal user named **ack_cis** in the Alibaba Cloud Linux 2 operating system for which CIS reinforcement is enabled.

CIS Level 1 Scored items that are covered by CIS reinforcement

CIS Aliyun Linux 2 Benchmark version 1.0.0 contains 145 Level 1 Scored items. Based on analysis and testing of these items, ACK provides CIS reinforcement for 128 out of the 145 items. The coverage is more than 88%.

CIS Level 1 Scored items that are not covered by CIS reinforcement

Item	Reason why the item is not covered by CIS reinforcement
1.1.2 Ensure /tmp is configured (Scored)	Involves partition modifications.
1.1.18 Ensure sticky bit is set on all world-writable directories (Scored)	Affects the control logic of ACK.
1.7.1.1 Ensure message of the day is configured properly (Scored)	Requires the deletion of the link to the user guide in the Message of the Day (MOTD) of Alibaba Cloud Linux 2 operating system.
3.1.1 Ensure IP forwarding is disabled (Scored)	Affects the networking component of ACK.
3.5.1.1 Ensure default deny firewall policy (Scored)	Requires the configuration of firewall policies.
3.5.1.2 Ensure loopback traffic is configured (Scored)	Requires the configuration of loopback rules.
3.5.1.4 Ensure firewall rules exist for all open ports (Scored)	Requires the configuration of firewall rules for open ports.
3.5.2.1 Ensure IPv6 default deny firewall policy (Scored)	Requires the configuration of IPv6 firewall policies.
3.5.2.2 Ensure IPv6 loopback traffic is configured (Scored)	Requires the configuration of IPv6 loopback rules.
4.2.1.4 Ensure rsyslog is configured to send logs to a remote log host (Scored)	Requires the configuration of rsyslog to send log data to a remote log host.
4.2.3 Ensure permissions on all logfiles are configured (Scored)	Requires the modification of a large number of files, which results in potential security risks.
5.2.10 Ensure SSH root login is disabled (Scored)	Requires the creation of other accounts for authentication or the use of non-SSH connections, such as Virtual Network Computing (VNC) connections.
5.2.18 Ensure SSH access is limited (Scored)	Requires the configuration of users and groups that are allowed to access the system by using SSH.

Item	Reason why the item is not covered by CIS reinforcement
5.2.3 Ensure permissions on SSH private host key files are configured (Scored)	The GID of `ssh_keys` is hard-coded to 998 in the scan script. However, the GID may not be 998 in the system. The GID may be 996.
5.3.2 Ensure lockout for failed password attempts is configured (Scored)	The Benchmark configuration recommendations are quite different from the configuration file of the Alibaba Cloud Linux 2 system. We recommend that you proceed with caution.
6.1.11 Ensure no unowned files or directories exist (Scored)	Affects the control logic of ACK.
6.1.12 Ensure no ungrouped files or directories exist (Scored)	Affects the control logic of ACK.

You can refer to the following sections in CIS Aliyun Linux 2 Benchmark version 1.0.0 to add fixes for CIS Level 1 Scored items that are not covered by CIS reinforcement. You can add fixes based on the **Remediation** section and check whether the fix works as expected based on the **Audit** section.

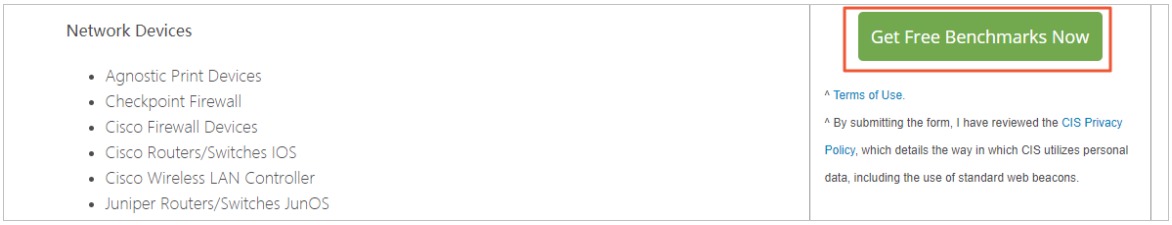
Section	Description
Profile Applicability	Whether the item belongs to Level 1 or Level 2.
Decription	The brief introduction of the item.
Rationale	The details and background information about the item. This helps you understand the reason for the recommended reinforcement.
Audit	The command script that is used to check whether the system meets the criteria. You can determine whether reinforcement is required based on the return value of the script.
Remediation	If the script in the Audit section indicates that reinforcement is required, you can run this script to reinforce the system.
Impact	Possible impacts if the system is not properly configured.
References	References.
CIS Controls	The description of the CIS control that corresponds to the item. To download CIS Controls, you must create an account.

Download CIS Aliyun Linux 2 Benchmark version 1.0.0

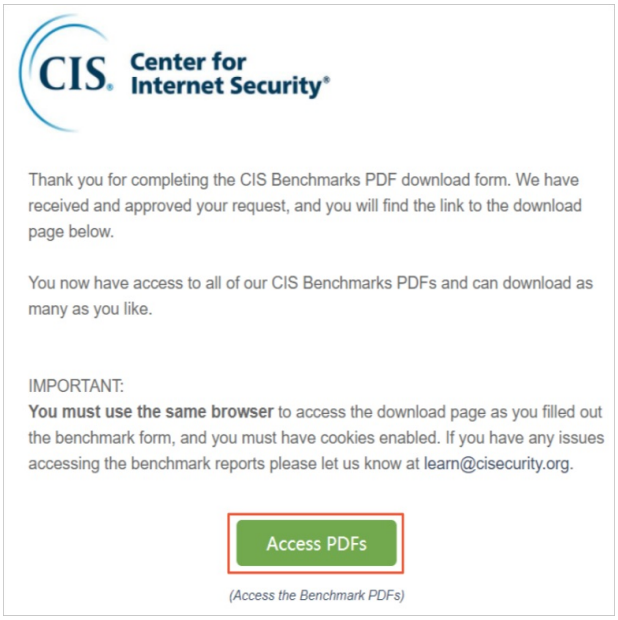
1. Log on to the CIS Benchmark homepage.
2. Select **Operating Systems** and **Linux**.

3. Find Aliyun Linux and click **Download CIS Benchmark**.

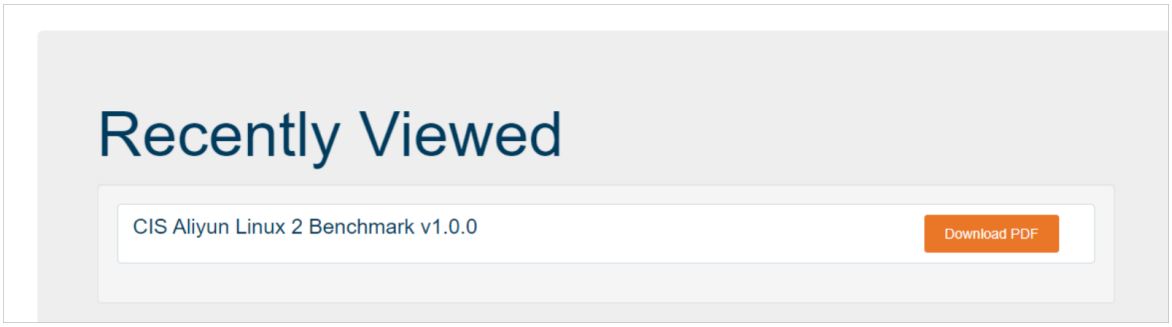
4. On the download page that appears, enter your basic information and click **Get Free Benchmarks Now**.



5. Wait a few minutes. Check your email inbox and find the email from CIS. Click **Access PDFs** in the email.



6. On the download page, find CIS Aliyun Linux 2 Benchmark v1.0.0 and click **Download PDF**.



Use CIS-CAT to evaluate the compliance of an ACK cluster with the CIS Benchmark

To evaluate the compliance of an ACK cluster with the CIS Benchmark, you can use CIS-Configuration Assessment Tool (CAT) to scan the cluster. CIS-CAT is a configuration assessment tool that scans the configuration of a system to provide a detailed evaluation report. You can run this tool on a system to obtain a benchmark score against a specified CIS Benchmark profile. The tool also provides remediation steps for noncompliant configurations. For more information, see CIS-CAT.

CIS-CAT has two editions: Lite and Pro. CIS-CAT Lite provides limited features and supports only the following systems: Windows 10, Ubuntu 18.04, and Google Chrome. CIS-CAT Lite does not support Alibaba Cloud Linux 2 and therefore cannot be used to scan ACK clusters for compliance evaluation.

CIS-CAT Pro has two versions: v4 and v3. The following section shows how to use CIS-CAT Pro v4 to scan an ACK cluster to evaluate the compliance of the cluster with the CIS Benchmark.

1. Go to CIS SecureSuite and register a CIS SecureSuite membership. Then, download the CIS-CAT Pro installation package named Assessor-CLI-v4.0.23.zip.
2. Log on to a cluster node that runs Alibaba Cloud Linux 2.

   For more information about how to connect to an Elastic Compute Service (ECS) node in an ACK cluster, see View nodes and Overview.
3. Run the following commands in sequence to install a Java environment that is required by CIS-CAT:

   ```
   yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel
   ```

   ```
   cat > /etc/profile.d/java8.sh <<EOF
   export JAVA_HOME=$(dirname $(dirname $(readlink $(readlink $(which javac)))))
   export PATH=$PATH:$JAVA_HOME/bin
   export CLASSPATH=.:$JAVA_HOME/jre/lib:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
   EOF
   ```

   ```
   source /etc/profile.d/java8.sh
   ```
4. Run the following commands in sequence to use CIS-CAT Pro to scan the node:

   ```
   unzip Assessor-CLI-v4.0.23.zip
   ```

   ```
   cd Assessor-CLI
   ```

   ```
   chmod +x ./Assessor-CLI.sh
   ```

   ```
   ./Assessor-CLI.sh  -b ./benchmarks/CIS_Aliyun_Linux_2_Benchmark_v1.0.0-xccdf.xml  -p "Level 1" -html
   ```

   > **Note**
   > - `-b` : specifies the benchmark based on which the node is scanned. The parameter value includes the operating system and benchmark version.
   > - `-p` : specifies the level of items that are scanned. In this example, Level 1 is specified because only CIS Level 1 Scored items need to be scanned.
5. Check the scan result.

```
--------------------------------------------------------------------------------
***** Assessment Results Summary *****
--------------------------------------------------------------------------------
  Total # of Results: 204
Total Scored Results: 145
          Total Pass: 128
          Total Fail: 17
         Total Error: 0
       Total Unknown: 0
Total Not Applicable: 0
   Total Not Checked: 11
  Total Not Selected: 36
 Total Informational: 12
--------------------------------------------------------------------------------
***** Assessment Scoring *****
--------------------------------------------------------------------------------
        Score Earned: 128.0
   Maximum Available: 145.0
               Total: 88.28%
--------------------------------------------------------------------------------
```

The following table describes the parameters in the scan result. For more information, see CIS-CAT Pro Assessor v4 Report.

Parameter	Description
Total # of Results	The total number of items that are provided by the specified benchmark. CIS Aliyun Linux 2 Benchmark v1.0.0 contains 204 items.
Total Scored Results	The total number of Scored items that belong to the specified level. Level 1 contains 145 items.
Total Pass	The total number of Scored items that belong to the specified level and passed the check. ACK provides CIS reinforcement for 128 Level 1 Scored items.
Total Fail	The total number of Scored items that belong to the specified level and failed the check. ACK does not provide CIS reinforcement for 17 Level 1 Scored items.
Total Error	The total number of Scored items that belong to the specified level and caused errors during script execution. In this example, no error occurred and therefore the result is 0.
Total Unknown	The total number of Scored items that belong to the specified level and where CIS-CAT was unable to determine if the criteria were met. In this example, the result is 0.
Total Not Applicable	The total number of items of the specified benchmark that are not applicable to the operating system. When you use CIS-CAT Pro to scan a node that runs Alibaba Cloud Linux 2 against CIS Aliyun Linux 2 Benchmark v1.0.0, all items apply.

Parameter	Description
Total Not Checked	These items are Not Scored. The items that belong to the Total Informational category are also Not Scored.
Total Not Selected	The total number of items of the specified benchmark that are not checked. In this example, CIS-CAT Pro checks only Level 1 items. Therefore, the 36 Level 2 items are not checked.
Total Informational	The total number of items that require manual evaluation. These items are Not Scored in the specified level.

Related information

- Overview
- CIS Aliyun Linux 2 Benchmark version 1.0.0
- CIS-CAT
- CIS-CAT Pro Assessor v4 Report

3.2.2. Use Alibaba Cloud Linux 2

Container Service for Kubernetes (ACK) allows you to create nodes that run the Alibaba Cloud Linux 2 operating system. These nodes leverage the high-performance kernel of Alibaba Cloud Linux 2 to provide optimized solutions to various scenarios. This topic describes the benefits of using Alibaba Cloud Linux 2 in ACK clusters and the optimizations of Alibaba Cloud Linux 2 that are provided by ACK to meet the requirements in different scenarios.

Context

Alibaba Cloud Linux 2 is a next-generation proprietary Linux distribution developed by Alibaba Cloud. This operating system provides a safe, stable, and high-performance customized environment for cloud applications. Alibaba Cloud Linux 2 is optimized for the cloud infrastructure and is designed to deliver a better runtime experience. Alibaba Cloud Linux 2 images are free of charge. Alibaba Cloud also provides long-term technical support (LTS) for Alibaba Cloud Linux 2.

Benefits

Alibaba Cloud Linux 2 is developed to make the most of the Alibaba Cloud IaaS platform, and is integrated with various optimizations and features:

- The Linux distribution with the fastest boot time on Alibaba Cloud.
- Deeply optimized for high-specification virtual machines and bare metal servers. This is ideal for multi-task scenarios that involve large-sized instances.
- Pre-installed with commonly used management software, such as Alibaba Cloud CLI and cloud-init. This reduces the costs of cloud resource management.
- A streamlined kernel that minimizes the attack surface and system resource occupation.
- An optimized technical support system that offers multi-channel technical support on Alibaba Cloud.
- Provides fixes for vulnerabilities at the earliest opportunity based on Common Vulnerabilities and Exposures (CVE).
- Supports live patching of the kernel. This ensures service continuity when vulnerabilities are being fixed.

Alibaba Cloud Linux 2 provides the following performance benefits for instances that run Alibaba Cloud Linux 2:

- Significantly reduces the system boot time of ECS instances, scales out computing resources when system overloading is detected, and reduces the system boot time by 29% compared with CentOS 7.
- Provides multitasking optimizations for ECS instances and improves the performance of large-sized instances by 16%.
- Improves the efficiency of system scheduling by 11%.
- Optimizes the Linux networking stack, providing improved network performance of 7.8% compared with CentOS 7.
- Improves bandwidth stability in scenarios in which frequent Internet access is required. Alibaba Cloud Linux 2 allows you to change the congestion control algorithm of containers to the Bottleneck Bandwidth and Round-trip Propagation Time (BBR) congestion control algorithm. All Alibaba Cloud Linux 2 images come with BBR compiled.
- Optimizes encryption based on the TLS protocol.
- Supports the Budget Fair Queueing (BFQ) I/O scheduler to reduce disk latency.

Optimizations on ACK in different scenarios

Alibaba Cloud provides kernel optimizations to allow containerized workloads to support more container tasks without service interruptions. ACK offers other optimizations to improve the speed and stability of containerized workloads in various scenarios. These optimizations are based on the optimizations provided by Alibaba Cloud Linux 2 and the kernel of this operating system.

- **IP Virtual Server (IPVS)**
  - Scenario 1: IPVS mode is enabled on high-specification instances that have more than 64 vCPUs and a large number of virtual IP addresses
    Problem: In IPVS mode, the estimation timer periodically calculates the transmission rate of each connection. When a large number of connections are established, this operation can occupy the CPU for a long period of time. This delays packet reception and results in a **ping** time of 200 ms.
    Solution: Schedule the IPVS estimation timer to a worker node and add a **sysctl** command to disable the IPVS estimation timer.
    Effect: The latency caused by the estimation timer no longer exists.
  - Scenario 2: Perform a rolling update
    Problem: During a rolling update, if the 5-tuple is unchanged and a new TCP SYN packet matches the connection record in the original IPVS 5-tuple, IPVS drops the SYN packet. However, the SYN packet needs to be sent to the new destination address. This results in SYN packet retransmission and a 1s latency.
    Solution: If new conntrack entries exist, release the connections that are in the TIME_WAIT state in the conntrack table and replace the connections with new connections.
    Effect: Network traffic can be switched to real servers almost without latency.

- **CoreDNS**
  Scenario 1: A large number of DNS queries result in a full conntrack table
  Problem: When applications send DNS requests to query static addresses or ports, the conntrack entry changes to the stream mode. DNS queries use the UDP protocol. UDP is stateless, fast, and requires no prior communications to establish connections. This creates a large number of UDP conntrack entries that are no longer needed. If these UDP conntrack entries are not deleted in a timely manner, the conntrack table may be full. This degrades NAT performance.
  Solution:
  - Conntrack entries are set to stream mode if the UDP connections last more than 2s. This avoids the rapid increase of conntrack entries.

- Shorten the validity duration of UDP conntrack from 180s to 120s. This reduces the impact of UDP requests on conntrack entries.

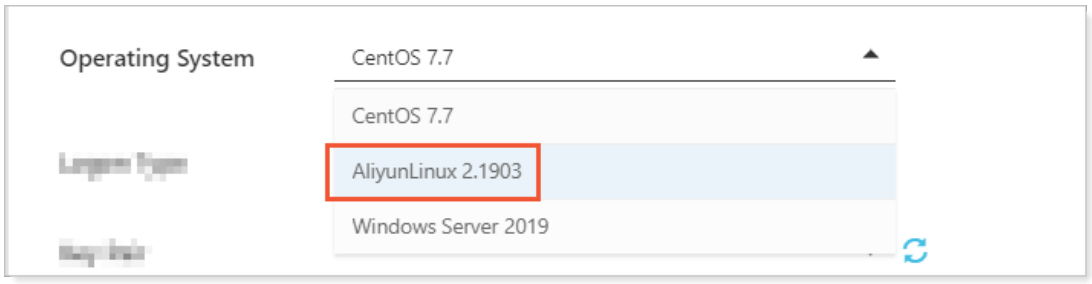
Effect: In the testing environment, the number of UDP conntrack entries is reduced by 50%.

- **Container networking**
  The Terway network plug-in supports the IPVLAN driver, which improves the network performance in short-packet communications by 40% compared with the traditional bridge interface and policy-based routing (PBR). By default, the BBR congestion control algorithm is compiled in Alibaba Cloud Linux 2. In scenarios in which frequent Internet access is required, you can change the congestion control algorithm of containers to BBR to improve the bandwidth stability of Internet access. This significantly improves the efficiency when containers communicate with the Internet or ACK pulls images from the Internet.
- **Container security**
  Alibaba Cloud has established partnerships with the Kata Containers and Clear Linux communities. You can deploy the Kata Containers solution on ECS bare metal instances. ACK also provides optimizations to reduce the boot time of runV containers. This ensures that the Kata Containers solution can work as expected. ACK also provides clusters that run sandboxed containers. This type of cluster offers similar user experience as normal clusters. You can deploy applications in lightweight sandboxed environments to isolate the workloads of multiple tenants. You can also use sandboxed containers to isolate untrusted applications. Sandboxed-Container improves security and has little impact on application performance.
- **AutoScaler**
  Alibaba Cloud Linux 2 reduces the system boot time of ECS instances. The average system boot time is reduced by 60% compared with CentOS 7. When the system is overloaded, the auto scaling feature can scale out ECS instances to create more ACK clusters. Then, ACK schedules and starts application instances. Alibaba Cloud Linux 2 also provides quick scale-out of computing resources to handle unexpected traffic spikes.
- **Resource monitoring and control**
  The kernel of Alibaba Cloud Linux 2 provides fine-grained resource visualization and control capabilities for containers. The capabilities include PSI pressure metrics, per-cgroup kswapd, and memory priority. In ACK clusters that run Alibaba Cloud Linux 2, you can use CGroup Controllers to adopt the capabilities, implement fine-grained resource configuration, and perform on-the-fly tweaks on top of tools such as BufferIO Control, TCP, CPUSet, Mem, and NUMA. This provides a step-by-step improvement in resource utilization and minimizes the interference between applications.
- **AI and data acceleration**
  Alibaba Cloud Linux 2 enables large-sized instances to handle high-performance computing tasks much faster. This operating system also optimizes streaming reads and writes to improve the read and write efficiency for large model files. This significantly accelerates AI-assisted and high-performance computing. The following data is recorded in the staging environment:
  - Test: Use Alluxio to load 1,152 files that contain up to a total of 144 GB Object Storage Service (OSS) data over 64 threads. A CentOS instance requires 3 minutes and 25 seconds to complete this operation. In contrast, an Alibaba Cloud Linux 2 instance only 2 minutes and 19.037 seconds to complete this operation. This indicates that Alibaba Cloud Linux 2 is 1.6 times faster than CentOS.
  - Test: Train the ResNet50 model that has a batch size of 128 and cache the data to Alluxio. A CentOS instance equipped with a NVIDIA V100 GPU can achieve a speed of 5,212.00 images/s. In contrast, an Alibaba Cloud Linux 2 instance equipped with a NVIDIA V100 GPU can achieve a speed of 8,746.59 images/s. Alibaba Cloud Linux 2 is 1.7 times faster than CentOS.
- **Resource visibility**
  Multiple containers can reside on a host server and have direct access to the resources on the host. This results in applications competing for resources. Alibaba Cloud Linux 2 provides optimizations for the cgroup feature of the kernel to improve resource visibility and provide information about the resource usage of individual containers. Resource information such as that from commands such as top, cpuinfo and meminfo are provided to help streamline resource observation and planning.

- Others
  - Alibaba Cloud Linux 2 is built on Linux kernel 4.19. ACK integrates Alibaba Cloud Linux 2 with core capabilities and containerization best practices of Alibaba Group.
  - Reduces the performance loss of OverlayFS and minimizes the loss caused by containerization in storage.
  - In most scenarios, sysctls are namespaced. In Kernel 4.19, most sysctls can be separately configured on pods. This allows you to configure different TCP timeout values and retransmit timeout values for different applications. You cannot modify these parameters in CentOS 7. In Alibaba Cloud Linux 2, you can configure these parameters on pods.

Set the node operating system to Alibaba Cloud Linux 2

When you create an ACK cluster in the ACK console, set **Operating System** to **Alibaba Cloud Linux 2.1903**. This way, you can use Alibaba Cloud Linux 2 as the operating system image of the nodes in the cluster. For more information, see Create a dedicated Kubernetes cluster.



> **Note**
> After you select Alibaba Cloud Linux 2, ACK automatically checks for security patches of Alibaba Cloud Linux 2 and installs the patches when you create the cluster, expand the cluster, or scale the cluster by adding or removing nodes to or from the cluster.

Related information

-
- Alibaba Cloud Linux 2 product page
-
- Alibaba Cloud Kernel official website
-
-

3.3. Create a cluster

3.3.1. Create a dedicated Kubernetes cluster

A dedicated Kubernetes cluster contains at least three master nodes. This ensures high availability and provides fine-grained management on clusters. You must manually size, maintain, and upgrade dedicated Kubernetes clusters. This topic describes how to create a dedicated Kubernetes cluster in the Container Service for Kubernetes (ACK) console.

Prerequisites

Resource Access Management (RAM) is activated in the RAM console. Auto Scaling (ESS) is activated in the ESS console.

Note
To use an ACK cluster, take note of the following limits:

- Server Load Balancer (SLB) instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
- ACK clusters support only virtual private clouds (VPCs).
- By default, you can create only a limited amount of cloud resources with each Alibaba Cloud account. You cannot create clusters if the quotas are exhausted. To request a quota increase, submit a ticket.
  - By default, you can create up to 50 clusters across all regions with each account. Each cluster can contain up to 100 nodes. To increase the quota of clusters or nodes,submit a ticket.

    Notice By default, you can add up to 48 route entries to the VPC where an ACK cluster is deployed. This means that you can configure up to 48 route entries for ACK clusters deployed in a VPC. To increase the quota of route entries for a VPC, submit a ticket.
  - By default, you can create up to 100 security groups with each account.
  - By default, you can create up to 60 pay-as-you-go SLB instances with each account.
  - By default, you can create up to 20 elastic IP addresses (EIPs) with each account.
- Limits on Elastic Compute Service (ECS) instances:
  The pay-as-you-go and subscription billing methods are supported.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Create Kubernetes Cluster**.
4. Click the **Dedicated Kubernetes** tab and configure the cluster.
   i. Configure basic settings of the cluster.

Parameter	Description
Cluster Name	Enter a name for the ACK cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
Region	Select a region to deploy the cluster.
Time Zone	Select a time zone for the ACK cluster. By default, the time zone configured for your browser is selected.

Parameter	Description
All Resources	Move the pointer over **All Resources** at the top of the page and select the resource group that you want to use. After you select a resource group, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource group. When you create a cluster, only the VPCs and vSwitches that belong to the selected resource group are displayed in the console. 
Kubernetes Version	The Kubernetes versions that are supported by ACK.
Container Runtime	The **containerd**, **Docker**, and **Sandboxed-Container** runtimes are supported. For more information, see Comparison of Docker, containerd, and Sandboxed-Container.
VPC	Select a VPC to deploy the cluster. Standard VPCs and shared VPCs are supported. ■ Shared VPC: The owner of a VPC (resource owner) can share the vSwitches in the VPC with other accounts in the same organization. ■ Standard VPC: The owner of a VPC (resource owner) cannot share the vSwitches in the VPC with other accounts. **Note** ACK clusters support only VPCs. You can select a VPC from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.
VSwitch	Select vSwitches. You can select up to three vSwitches that are deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.

Parameter	Description
Network Plug-in	Select a network plug-in. Flannel and Terway are supported. For more information, see Terway and Flannel. ■ Flannel: a simple and stable Container Network Interface (CNI) plug-in that is developed by open source Kubernetes. Flannel provides a few simple features. However, it does not support standard Kubernetes network policies. ■ Terway: a network plug-in that is developed by ACK. Terway allows you to assign elastic network interfaces (ENIs) of Alibaba Cloud to containers. It also allows you to customize Kubernetes network policies to regulate how containers communicate with each other and implement bandwidth throttling on individual containers. **Note** ■ The number of pods that can be deployed on a node depends on the number of ENIs that are attached to the node and the maximum number of secondary IP addresses that are provided by these ENIs. ■ If you select a shared VPC for an ACK cluster, you must select Terway as the network plug-in. ■ If you select **Terway**, an ENI is shared among multiple pods. A secondary IP address of the ENI is assigned to each pod.
Pod CIDR Block	If you select **Flannel** as the network plug-in, you must set **Pod CIDR Block**. The CIDR block specified by **Pod CIDR Block** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.

Parameter	Description
Terway Mode	If you set **Network Plug-in** to **Terway**, you must set **Terway Mode**. ■ Select or clear **Assign One ENI to Each Pod**. ■ If you select the check box, an ENI is assigned to each pod. ■ If you clear the check box, an ENI is shared among multiple pods. A secondary IP address that is provided by the ENI is assigned to each pod. **Note** To select the **Assign One ENI to Each Pod** check box, you must submit a ticket to apply to be added to a whitelist. ■ Select or clear **IPVLAN**. ■ This option is available only when you clear Assign One ENI to Each Pod. ■ If you select IPVLAN, IPVLAN and extended Berkeley Packet Filter (eBPF) are used for network virtualization when an ENI is shared among multiple pods. This improves network performance. Only the Alibaba Cloud Linux 2 operating system is supported. ■ If you clear IPVLAN, policy-based routes are used for network virtualization when an ENI is shared among multiple pods. The CentOS 7 and Alibaba Cloud Linux 2 operating systems are supported. This is the default setting. For more information about the IPVLAN feature in Terway mode, see Terway IPvlan. ■ Select or clear **Support for NetworkPolicy**. ■ The NetworkPolicy feature is available only when you clear Assign One ENI to Each Pod. By default, Assign One ENI to Each Pod is unselected. ■ If you select Support for NetworkPolicy, you can use Kubernetes network policies to control the communication among pods. ■ If you clear Support for NetworkPolicy, you cannot use Kubernetes network policies to control the communication among pods. This prevents Kubernetes network policies from overloading the Kubernetes API server.
Service CIDR	Set **Service CIDR**. The CIDR block specified by **Service CIDR** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.

Parameter	Description
IP Addresses per Node	If you select **Flannel** as the network plug-in, you must set **IP Addresses per Node**. Note ■ **IP Addresses per Node** specifies the maximum number of IP addresses that can be assigned to each node. We recommend that you use the default value. ■ After you select the VPC and specify the number of IP addresses per node, recommended values are automatically generated for Pod CIDR block and Service CIDR block. The system also provides the maximum number of nodes that can be deployed in the cluster and the maximum number of pods that can be deployed on each node. You can modify the values based on your business requirements.
Configure SNAT	By default, an ACK cluster cannot access the Internet. If the VPC that you select for the ACK cluster cannot access the Internet, you can select **Configure SNAT for VPC**. This way, ACK will create a NAT gateway and configure Source Network Address Translation (SNAT) rules to enable Internet access for the VPC.
Access to API Server	By default, an internal-facing Server Load Balancer (SLB) instance is created for the cluster API server. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. Notice If you delete the SLB instance, you cannot access the cluster API server. Select or clear **Expose API Server with EIP**. The ACK API server provides multiple HTTP-based RESTful APIs, which can be used to create, delete, modify, query, and monitor resources, such as pods and Services. ■ If you select this check box, an elastic IP address (EIP) is created and associated with an Internet-facing SLB instance. Port 6443 used by the API server is opened on master nodes. You can connect to and manage the ACK cluster by using kubeconfig over the Internet. ■ If you clear this check box, no EIP is created. You can connect to and manage the ACK cluster by using kubeconfig only within the VPC.

Parameter	Description
SSH Logon	To enable Secure Shell (SSH) logon, you must first select **Expose API Server with EIP**. ■ If you select this check box, you can access the cluster by using SSH. ■ If you clear this check box, you cannot access the cluster by using SSH or kubectl. If you want to access an ECS instance in the cluster by using SSH, you must manually associate an EIP with the instance and configure security group rules to open SSH port 22. For more information, see Use SSH to connect to an ACK cluster.
RDS Whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the cluster to the RDS whitelist. **Note** To enable an RDS instance to access the cluster, you must deploy the RDS instance in the VPC where the cluster is deployed.
Security Group	You can select **Create Basic Security Group**, **Create Advanced Security Group**, or **Select Existing Security Group**. For more information, see Overview. **Note** To select **Select Existing Security Group**, Submit a ticket to apply to be added to a whitelist.

ii. Configure advanced settings of the cluster.

Parameter	Description
Kube-proxy Mode	iptables and IPVS are supported. ■ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the ACK cluster. This mode is suitable for ACK clusters that manage a small number of Services. ■ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for ACK clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.

Parameter	Description
Labels	Add labels to the cluster. Enter a key and a value, and click **Add**. **Note** ■ Key is required. *Value* is optional. ■ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ■ *Values* are not case-sensitive. A value can be empty and can contain up to 128 characters in length. It cannot be http:// or https://. ■ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the others that use the same key. ■ If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Custom Image	You can select a custom image for your ECS nodes. After you select a custom image, all nodes in the cluster are deployed by using this image. For more information about how to create a custom image, see Create a Kubernetes cluster by using a custom image. **Note** ■ Only custom images based on CentOS 7.x and Alibaba Cloud Linux 2.x are supported. ■ To use this feature, submit a ticket to apply to be added to a whitelist.
Cluster Domain	Set the domain name of the cluster. **Note** The default domain name is **cluster.local**. You can enter a custom domain name. A domain name consists of two parts. Each part must be 1 to 63 characters in length and can contain only letters and digits. You cannot leave these parts empty.
Custom Certificate SANs	You can enter custom subject alternative names (SANs) for the API server certificate of the cluster to accept requests from specified IP addresses or domain names.
Service Account Token Volume Projection	**Service account token volume projection** reduces security risks when pods use service accounts to access the API server. This feature enables kubelet to request and store the token on behalf of the pod. This feature also allows you to configure token properties, such as the audience and validity duration. For more information, see Enable service account token volume projection.

Parameter	Description
Cluster CA	If you select Custom Cluster CA, upload a Certificate Authority (CA) certificate for the cluster to ensure secure data transmission between the server and client.
Deletion Protection	

5. Click **Next: Master Configurations** to configure master nodes.

Parameter	Description
Billing Method	The **pay-as-you-go** and **subscription** billing methods are supported. If you select the **subscription** billing method, you must set the following parameters: ○ **Duration**: You can select 1, 2, 3, or 6 months. If you require a longer duration, you can select 1 to 5 years. ○ **Auto Renewal**: Specify whether to enable auto-renewal.
Master Node Quantity	Specify the number of master nodes. You can create three or five master nodes.
Instance Type	Select an instance type for the master nodes. For more information, see Instance families.
System Disk	By default, system disks are mounted to master nodes. Enhanced SSDs, SSDs, and ultra disks are supported. Note ○ You can select **Enable Backup** to back up disk data. ○ If you select an **enhanced SSD** as the system disk, you can set a **performance level** for the disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.

6. Click **Next: Worker Configurations** to configure worker nodes.
   i. Set worker nodes.
   - If you select **Create Instance**, you must set the parameters that are listed in the following table.

Parameter	Description
Instance Type	You can select multiple instance types. For more information, see Instance families.
Selected Types	The selected instance types are displayed.
Quantity	Specify the number of worker nodes (ECS instances) to be created.

Parameter	Description
System Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. **Note** ■ You can select **Enable Backup** to back up disk data. ■ If you select **enhanced SSD** as the system disk type, you can set a custom **performance level** for the system disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.
Mount Data Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. You can enable **disk encryption** and **disk backup** when you mount a data disk.
Operating System	ACK supports the following node operating systems: ■ Alibaba Cloud Linux 2. This is the default operating system. If you select Alibaba Cloud Linux 2, you can configure security reinforcement for the operating system: ■ Disable: disables security reinforcement for Alibaba Cloud Linux 2.x. ■ CIS Reinforcement: enables security reinforcement for Alibaba Cloud Linux 2.x. For more information about CIS reinforcement, see CIS reinforcement. ■ CentOS 7.x **Note** CentOS 8.x and later are not supported.
Logon Type	■ Key pair logon ■ **Key Pair**: Select an SSH key pair from the drop-down list. ■ **create a key pair**: Create an SSH key pair if none is available. For more information about how to create an SSH key pair, see Create an SSH key pair. After the key pair is created, set it as the credential that is used to log on to the cluster. ■ Password logon ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.

■ If you select **Add Existing Instance**, make sure that you have created ECS instances in the region where the cluster is deployed. Then, set **Operating System**, **Logon Type**, and **Key Pair** based on the preceding settings.

ii. Configure advanced settings of worker nodes.

Parameter	Description
Node Protection	Specify whether to enable node protection. **Note** By default, this check box is selected. Node protection prevents nodes from being accidentally deleted in the console or by calling the API. This prevents user errors.
User Data	For more information, see Overview of ECS instance user data.
Custom Node Name	Specify whether to use a **custom node name**. A node name consists of a prefix, an IP substring, and a suffix. ■ Both the prefix and suffix can contain one or more parts that are separated by periods (.).These parts can contain lowercase letters, digits, and hyphens (-), and must start and end with a lowercase letter or digit. ■ The IP substring length specifies the number of digits to be truncated from the end of the returned node IP address. Valid values: 5 to 12. For example, if the node IP address is 192.1xx.x.xx, the prefix is aliyun.com, the IP substring length is 5, and the suffix is test, the node name will be aliyun.com00055test.
Node Port Range	Set the node port range. The default port range is 30000 to 32767.
CPU Policy	Set the CPU policy. ■ none: This policy indicates that the default CPU affinity is used. This is the default policy. ■ static: This policy allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Taints	Add taints to the worker nodes in the ACK cluster.

7. Click **Next:Component Configurations** to configure components.

Parameter	Description
Ingress	Specify whether to install Ingress controllers. By default, **Install Ingress Controllers** is selected. For more information, see Ingress高级用法.
Volume Plug-in	Select a volume plug-in. FlexVolume and CSI are supported. An ACK cluster can be automatically bound to Alibaba Cloud disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets that are mounted to pods in the cluster. For more information, see Storage management-FlexVolume and Storage management-CSI.
Monitoring Agents	Specify whether to install the Cloud Monitor agent. By default, **Install CloudMonitor Agent on ECS Instance** is selected.

Parameter	Description
Log Service	Specify whether to enable Log Service. You can select an existing Log Service project or create one. By default, **Enable Log Service** is selected. When you create an application, you can enable Log Service through a few steps. For more information, see Collect log files from containers by using Log Service. By default, **Install node-problem-detector and Create Event Center** is selected. You can also specify whether to **create Ingress dashboards** in the Log Service console.
Workflow Engine	Specify whether to enable Alibaba Cloud Genomics Service (AGS). **Note** To use this feature, submit a ticket to apply to be added to a whitelist. ○ If you select this check box, the system automatically installs the AGS workflow plug-in when the system creates the cluster. ○ If you clear this check box, you must manually install the AGS workflow plug-in. For more information, see Introduction to AGS CLI.

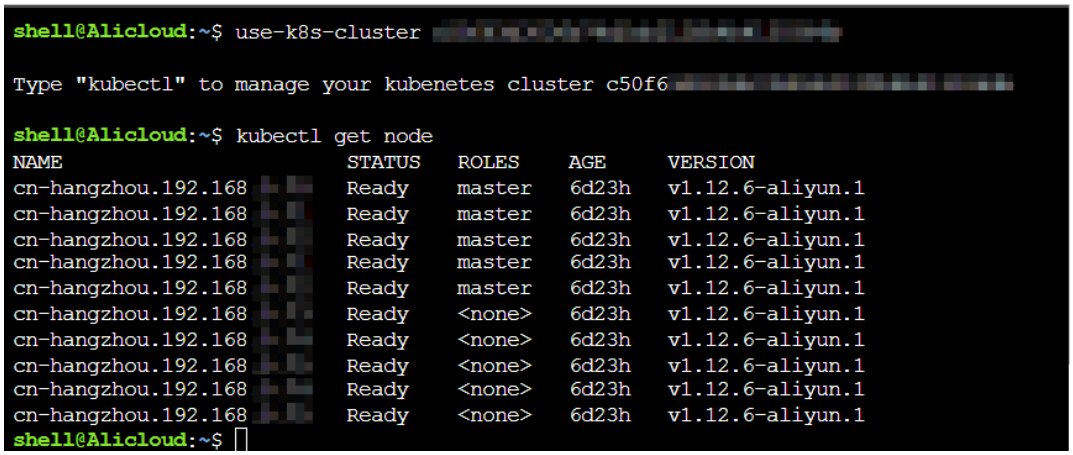
8. Click **Next: Confirm Order**.
9. Read Terms of Service, select the check box, and then click **Create Cluster**.

Note It requires about 10 minutes to create a dedicated Kubernetes cluster that contains multiple nodes.

Result

- After the cluster is created, you can view the newly created cluster on the **Clusters** page in the ACK console.
- Click **View Logs** in the Actions column. On the page that appears, you can view the cluster log. To view detailed log information, click **Stack events**.
- On the Clusters page, find the newly created cluster and click **Details** in the **Actions** column. On the details page of the cluster, click the **Basic Information** tab to view basic information about the cluster and click the **Connection Information** tab to view information about how to connect to the cluster. The following information is displayed:
  - **API Server Public Endpoint**: the IP address and port that the API server uses to provide services over the Internet. It allows you to manage the cluster by using kubectl or other tools on the client.
  - **API Server Internal Endpoint**: the IP address and port that the API server uses to provide services within the cluster. The endpoint belongs to the SLB instance that is bound to the cluster. Three master nodes work as the backend servers of the SLB instance.
  - **Testing Domain**: the domain name that is used to test Services. The suffix of the domain name is `<cluster_id>.<region_id>.alicontainer.com`.

    Note To rebind the domain name, click **Rebind Domain Name**.
- You can Connect to Kubernetes clusters by using kubectl and run the `kubectl get node` command to view information about the nodes in the cluster.



3.3.2. Create a Kubernetes cluster by using a custom image

In the cloud-native era, an increasing number of users choose to migrate applications and business to the cloud. Different business scenarios have different requirements on the container platform. An important requirement is to create Kubernetes clusters by using custom container images. This topic describes how to create a Kubernetes cluster by using a custom image.

Prerequisites

Before you create a Kubernetes cluster by using a custom image, take note of the following limits on the custom images that are supported by Container Service for Kubernetes (ACK):

- We recommend that you use the latest base images provided by ACK. The base images of ACK can be used to create Kubernetes clusters and have passed the strict tests of the ACK technical team. Custom images used to create Kubernetes clusters must meet the following requirements:
  - Alibaba Cloud cloud-init can be installed. For more information, see Install cloud-init.
  - sshd server is enabled and the default port 22 is used.
  - Time synchronization is performed by using a Network Time Protocol (NTP) server provided by Alibaba Cloud.
- If you want to use custom images, Submit a ticket.

Context

ACK allows you to create Kubernetes clusters by using custom images. However, you may encounter the following issues when you create and package custom images:

- It is not efficient to manually create images.
- Records may be missing in the image change history. This makes it difficult to locate faults.
- You cannot verify whether the custom images meet the requirements of nodes in ACK clusters.

To solve the preceding issues, we launch the open source ack-image-builder project to help you create custom images that meet the requirements of nodes in ACK clusters.
The ack-image-builder project is developed based on the open source tool HashiCorp Packer, and provides default configuration templates and verification scripts that are used to create and verify custom images.

To create a custom image by using ack-image-builder, perform the following steps:

Procedure

1. Install Packer.

Download Packer from the official website. Make sure that the downloaded version is compatible with your node operating system. Then, install and verify Packer by following the installation instructions of Packer.
Run the following command. If the following output is returned, it indicates that Packer is installed.

```
packer version
```

```
Packer v1.4.1
```

2. Configure a Packer template.

   When you create a custom image by using Packer, you must create a template file in JSON format. In the template file, specify the image builder provided by Alibaba Cloud and the provisioner that are used to create and configure custom images.

```
{
 "variables": {
  "region": "cn-hangzhou",
  "image_name": "test_image{{timestamp}}",
  "source_image": "centos_7_06_64_20G_alibase_20190711.vhd",
  "instance_type": "ecs.n1.large",
  "access_key": "{{env `ALICLOUD_ACCESS_KEY`}}",
  "secret_key": "{{env `ALICLOUD_SECRET_KEY`}}"
 },
 "builders": [
  {
   "type": "alicloud-ecs",
   "access_key": "{{user `access_key`}}",
   "secret_key": "{{user `secret_key`}}",
   "region": "{{user `region`}}",
   "image_name": "{{user `image_name`}}",
   "source_image": "{{user `source_image`}}",
   "ssh_username": "root",
   "instance_type": "{{user `instance_type`}}",
   "io_optimized": "true"
  }
 ],
 "provisioners": [
  {
   "type": "shell",
   "scripts": [
    "scripts/updateKernel.sh",
    "scripts/reboot.sh",
    "scripts/cleanUpKerneles.sh",
    "config/default.sh",
    "scripts/updateDNS.sh",
    "scripts/verify.sh"
   ],
   "expect_disconnect": true
  }
 ]
}
```

Parameter	Description
access_key	The AccessKey ID that is used to create a custom image.

Parameter	Description
secret_key	The AccessKey secret that is used to create a custom image.
region	The region of the cloud resources that are temporarily used to create a custom image.
image_name	The name of the custom image.
source_image	The name of the base image used to create a custom image. You can obtain the name of a base image from the public image list of Alibaba Cloud.
instance_type	The type of the cloud resources that are temporarily used to create the custom image.
provisioners	The type of the provisioner used to create a custom image.

3. Create a Resource Access Management (RAM) user and create an AccessKey pair for the RAM user.

   We recommend that you create a RAM user and attach a RAM policy with Packer-related permissions to the RAM user. You must also create an AccessKey pair for the RAM user.

4. Add the AccessKey pair to the template and create a custom image by using the template.

   i. Run the following commands to add the AccessKey pair to the template:

   ```
   export ALICLOUD_ACCESS_KEY=XXXXXX
   export ALICLOUD_SECRET_KEY=XXXXXX
   ```

   ii. Run the following command to create a custom image by using the template:

   ```
   packer build alicloud.json
   ```

```
alicloud-ecs output will be in this color.
==> alicloud-ecs: Prevalidating source region and copied regions...
==> alicloud-ecs: Prevalidating image name...
  alicloud-ecs: Found image ID: centos_7_06_64_20G_alibase_20190711.vhd
==> alicloud-ecs: Creating temporary keypair: xxxxxx
==> alicloud-ecs: Creating vpc...
  alicloud-ecs: Created vpc: xxxxxx
==> alicloud-ecs: Creating vswitch...
  alicloud-ecs: Created vswitch: xxxxxx
==> alicloud-ecs: Creating security group...
  alicloud-ecs: Created security group: xxxxxx
==> alicloud-ecs: Creating instance...
  alicloud-ecs: Created instance: xxxxxx
==> alicloud-ecs: Allocating eip...
  alicloud-ecs: Allocated eip: xxxxxx
  alicloud-ecs: Attach keypair xxxxxx to instance: xxxxxx
==> alicloud-ecs: Starting instance: xxxxxx
==> alicloud-ecs: Using ssh communicator to connect: 47.111.127.54
==> alicloud-ecs: Waiting for SSH to become available...
==> alicloud-ecs: Connected to SSH!
==> alicloud-ecs: Provisioning with shell script: scripts/verify.sh
  alicloud-ecs: [20190726 11:04:10]: Check if kernel version >= 3.10. Verify Passed!
  alicloud-ecs: [20190726 11:04:10]: Check if systemd version >= 219. Verify Passed!
  alicloud-ecs: [20190726 11:04:10]: Check if sshd is running and listen on port 22. Verify Passed!
  alicloud-ecs: [20190726 11:04:10]: Check if cloud-init is installed. Verify Passed!
  alicloud-ecs: [20190726 11:04:10]: Check if wget is installed. Verify Passed!
  alicloud-ecs: [20190726 11:04:10]: Check if curl is installed. Verify Passed!
  alicloud-ecs: [20190726 11:04:10]: Check if kubeadm is cleaned up. Verify Passed!
  alicloud-ecs: [20190726 11:04:10]: Check if kubelet is cleaned up. Verify Passed!
  alicloud-ecs: [20190726 11:04:10]: Check if kubectl is cleaned up. Verify Passed!
  alicloud-ecs: [20190726 11:04:10]: Check if kubernetes-cni is cleaned up. Verify Passed!
==> alicloud-ecs: Stopping instance: xxxxxx
==> alicloud-ecs: Waiting instance stopped: xxxxxx
==> alicloud-ecs: Creating image: test_image1564110199
  alicloud-ecs: Detach keypair xxxxxx from instance: xxxxxxx
==> alicloud-ecs: Cleaning up 'EIP'
==> alicloud-ecs: Cleaning up 'instance'
==> alicloud-ecs: Cleaning up 'security group'
==> alicloud-ecs: Cleaning up 'vSwitch'
==> alicloud-ecs: Cleaning up 'VPC'
==> alicloud-ecs: Deleting temporary keypair...
Build 'alicloud-ecs' finished.
==> Builds finished. The artifacts of successful builds are:
--> alicloud-ecs: Alicloud images were created:
cn-hangzhou: m-bp1aifbnupnaktj00q7s
```

scripts/verify.sh specifies the verification result of the custom image.

5. Use the custom image to create an ACK cluster.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. In the upper-right corner of the **Clusters** page, click **Cluster Templates**.
   iv. In the **Select Cluster Template** dialog box, find **Standard Dedicated Cluster** and click **Create**. For more information, see Create a dedicated Kubernetes cluster.

v. Configure the basic settings and click **Show Advanced Options**. Click **Select** in the **Custom Image** section.

vi. In the **Choose Custom Image** dialog box, click **Use** on the right side of the custom image that you want to use.

vii. After the cluster configuration is completed, click **Create Cluster**.

After the cluster is created by using the custom image, the custom image is also used in other operations, such as cluster expansions.

3.4. Access clusters

3.4.1. Connect to Kubernetes clusters by using kubectl

You can connect to a Kubernetes cluster from your on-premises machine by using the kubectl command-line tool. This topic describes how to connect to a Container Service for Kubernetes (ACK) cluster by using kubectl.

Context

For more information about kubectl, see kubectl.

Connect to an ACK cluster by using kubectl

1. Install and set up a kubectl client.

   For more information, see Install and set up kubectl.

2. Configure the credentials used to access the cluster.

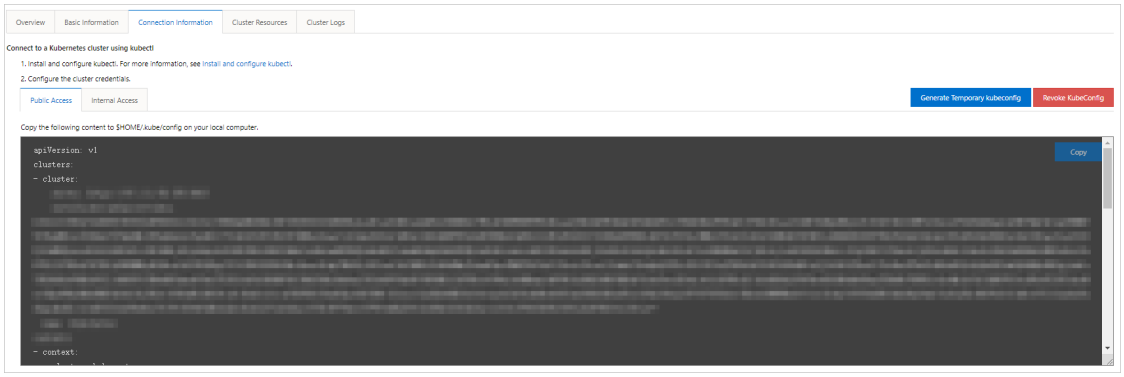
   i. Log on to the ACK console.

   ii. In the left-side navigation pane of the ACK console, click **Clusters**.

   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

   iv. In the left-side navigation pane of the details page, click **Cluster Information**.

   v. On the **Cluster Information** page, click the **Connection Information** tab.



vi. Configure the kubeconfig file used to access the cluster.

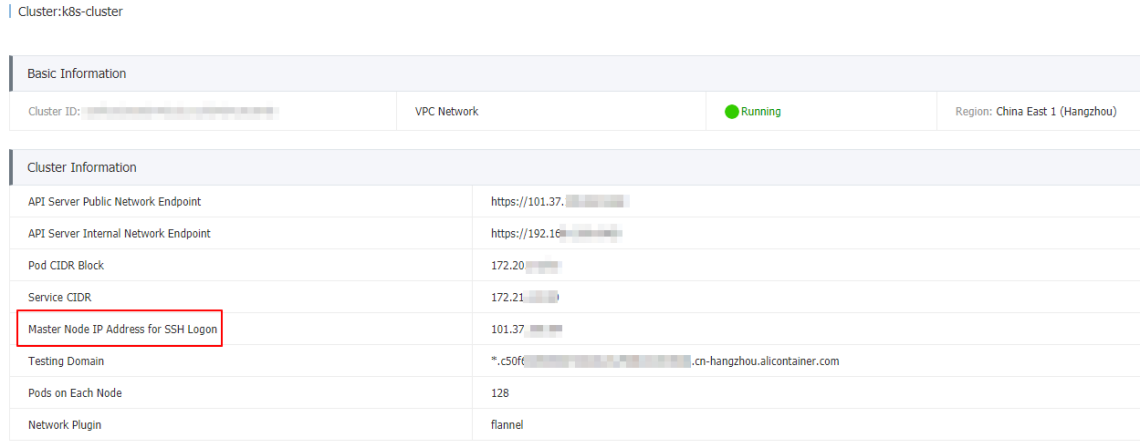
- To connect to the cluster over the Internet, click the **Public Access** tab and **copy** the content in the code block to the *$HOME/.kube/config* file of your on-premises machine.

  > **Note** By default, kubectl retrieves the *config* file in the *$HOME/.kube/config* directory. kubectl communicates with a Kubernetes cluster by using kubeconfig files.

- To connect to the cluster over the internal network, click the **Internal Access** tab and **copy** the content in the code block to the `$HOME/.kube/config` file of your on-premises machine.

Execution result

- If you want to connect to a managed Kubernetes cluster, you can run **kubectl** commands to connect to the cluster from your on-premises machine after you configure the kubeconfig file.
- If you want to connect to a dedicated Kubernetes cluster, you can obtain the IP address for SSH connection to a master node from the **Basic Information** tab and run **kubectl** commands to connect to the cluster from your on-premises machine.

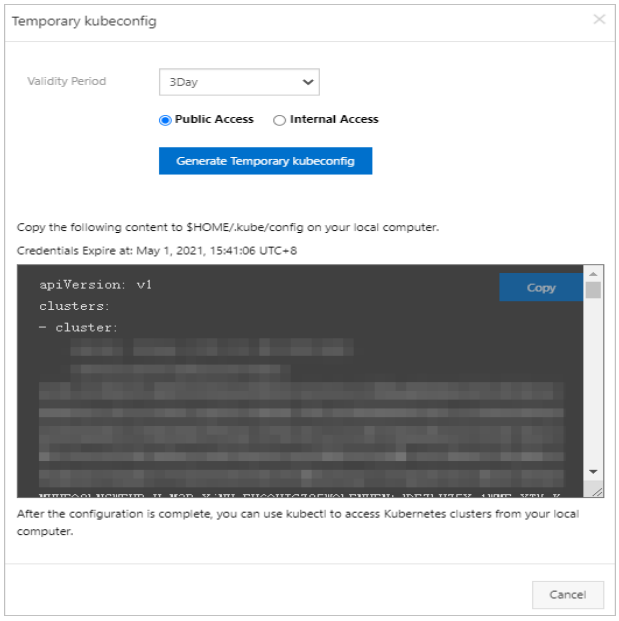


> **Note** To use SSH to log on to a dedicated Kubernetes cluster, you must enable **SSH logon** when you create the cluster. For more information, see Create a dedicated Kubernetes cluster.

Generate a temporary kubeconfig file

To ensure cluster security, you can generate a temporary kubeconfig file with a validity period. You can use this kubeconfig file to enable temporary access to ACK clusters.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the **Cluster Information** page, click the **Connection Information** tab, and then click **Generate Temporary kubeconfig**.
5. In the **Temporary kubeconfig** dialog box, set the validity period of the kubeconfig file and select the access mode (public access or internal access).



- If you want to connect to the cluster over the Internet, select **Public Access** and click **Generate Temporary kubeconfig**. Then, click **Copy** to copy the code block to the `$HOME/.kube/config` file of your on-premises machine. `kubectl` retrieves the credentials from this file.
- If you want to connect to the cluster over the internal network, select **Internal Access** and click **Generate Temporary kubeconfig**. Then, click **Copy** to copy the code block into the `$HOME/.kube/config` file of your on-premises machine. `kubectl` retrieves the credentials from this file.

Revoke a kubeconfig file

To revoke a kubeconfig file, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the **Cluster Information** page, click the **Connection Information** tab, and then click **Revoke KubeConfig**.
5. In the **Revoke KubeConfig** dialog box, click **OK**.

Related information

- DescribeClusterUserKubeconfig

3.4.2. Revoke a KubeConfig credential

In common multi-tenant scenarios, Alibaba Cloud Container Service for Kubernetes (ACK) issues KubeConfig credentials to users of different roles. These KubeConfig credentials contain unique identity information about the users and are used to connect to ACK clusters. If an employee leaves the company, you can revoke the KubeConfig credentials assigned to the employee. If a KubeConfig credential is suspected to be leaked, you can revoke the KubeConfig credential to ensure cluster security. This topic describes how to revoke a KubeConfig credential in the ACK console.

Prerequisites

- If you want to revoke a KubeConfig credential that is used to access a serverless Kubernetes cluster, select a cluster that was created after September 6, 2019.

- If you want to revoke a KubeConfig credential that is used to access a dedicated or managed Kubernetes cluster, pay attention to the following notes:
  - If you are using an Alibaba Cloud account, you can revoke your own KubeConfig credentials that are used to access clusters that were created no earlier than October 15, 2019.
  - If you are a RAM user, you can revoke KubeConfig credentials that are used to access clusters that you can manage.

KubeConfig credentials can be revoked in the following two scenarios:

- Revoke your own KubeConfig credentials.
- Log on to your Alibaba Cloud account to revoke KubeConfig credentials that are issued to RAM users.

Revoke your own KubeConfig credentials

> **Notice** After a KubeConfig credential is revoked, you cannot use the credential to connect to the cluster. Perform this operation with caution.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to view and click **Details** in the **Actions** column.

   > **Note**
   > If you are using an Alibaba Cloud account, select a cluster that was created no earlier than October 15, 2019.
   > If you want to revoke a KubeConfig credential that is used to access a serverless Kubernetes cluster, select a cluster that was created after September 6, 2019.

4. On the **Cluster Information** page, click the **Connection Information** tab, and then click **Revoke KubeConfig**.

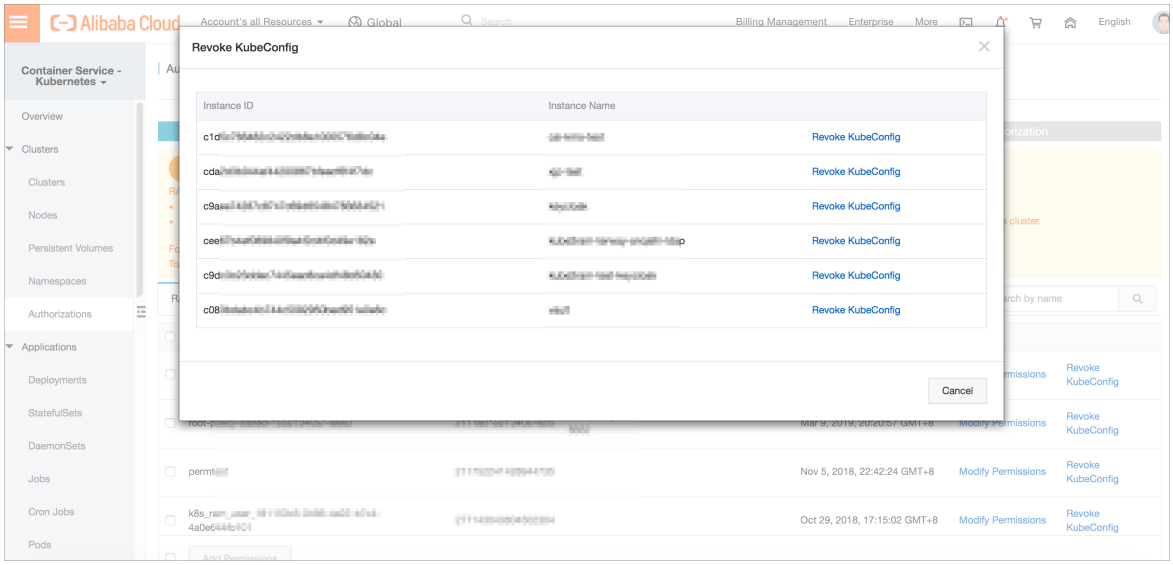
5. In the dialog box that appears, click **OK**.

   Your KubeConfig credential that is used to access the selected cluster is revoked. The system then automatically assigns you a new KubeConfig credential.

Use an Alibaba Cloud account to revoke a KubeConfig credential from a RAM user

1. Log on to the Container Service for Kubernetes console.
2. In the left-side navigation pane, choose **Authorizations**.
3. On the **RAM Users** tab of the **Authorizations** page, find the RAM user from which you want to revoke the KubeConfig credential and click **Revoke KubeConfig** on the right.

   In the dialog box that appears, you can find the IDs of all clusters that the selected RAM user can access.



4. Find the cluster for which you want to revoke the KubeConfig credential and click **Revoke KubeConfig**.
5. In the dialog box that appears, click **OK**.

3.4.3. Generate API request parameters

This topic describes how to generate API request parameters to create a Container Service for Kubernetes (ACK) cluster in the ACK console. When you fail to create an ACK cluster due to invalid parameter settings, you can use this feature to generate API request parameters. Then, you can create the ACK cluster by calling the API.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Cluster Template**.
4. In the **Select Cluster Template** dialog box, select the type of the cluster that you want to create and click **Create**.

   On the cluster configuration page that appears, set the parameters. For more information about the parameters, see Create a dedicated Kubernetes cluster.
5. Set the cluster parameters.

   To generate API request parameters, you must first specify a set of required parameters.

   - For more information about the parameters that are required to create a dedicated Kubernetes cluster, see Create a dedicated Kubernetes cluster.
   - For more information about the parameters that are required to create a managed Kubernetes cluster, see Create a managed Kubernetes cluster.
6. After you specify the required parameters, click **Generate API Request Parameters**.

   In the **API Request Parameters** dialog box, you can view the parameters that you have specified.
7. Click **Copy** to copy the cluster configurations.

   You can directly call the API to create an ACK cluster by using the copied cluster configurations. For more information, see Create a dedicated Kubernetes cluster and Create a managed Kubernetes cluster. You can also use the copied cluster configurations to create an ACK cluster by using Alibaba Cloud command-line interface (CLI). For more information, see Create a cluster.

3.4.4. Use kubectl on Cloud Shell to manage ACK clusters

This topic describes how to use kubectl on Cloud Shell to manage clusters of Container Service for Kubernetes (ACK) in the ACK console.

Prerequisites

创建Kubernetes托管版集群.

Context

To manage ACK clusters by using kubectl, you must install and set up the kubectl client. For more information, see Connect to Kubernetes clusters by using kubectl. You can also use kubectl on Cloud Shell in the ACK console to manage ACK clusters.

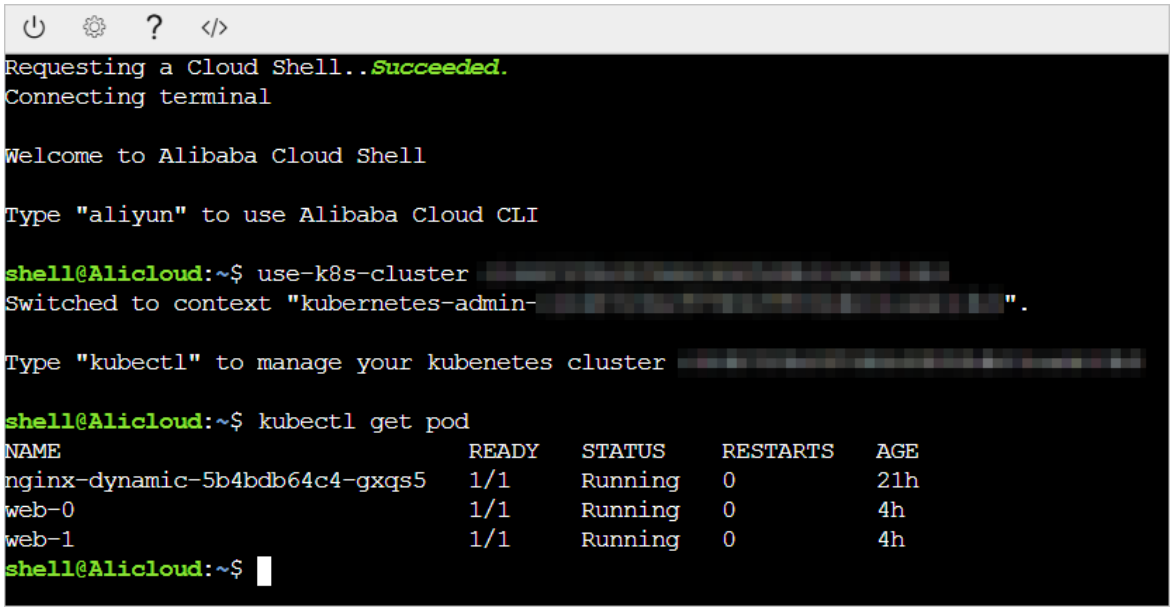
Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and choose **More > Open Cloud Shell** in the **Actions** column.

   > **Note**
   > After Cloud Shell is opened, perform the following steps:
   >
   > - On the **Authorization** page, click **OK**. You will obtain an AccessKey pair for a temporary session.
   > - Click the icon and select **Mount File Storage**. You can **create and mount** a Network Attached Storage (NAS) file system to the cluster or **skip** the process.

4. Use kubectl on Cloud Shell in the ACK console to manage ACK clusters.

   > **Note** When you open Cloud Shell, Cloud Shell automatically reads the *kubeconfig* file of the cluster. Then, you can use kubectl to manage the cluster.



3.4.5. Use SSH to connect to an ACK cluster

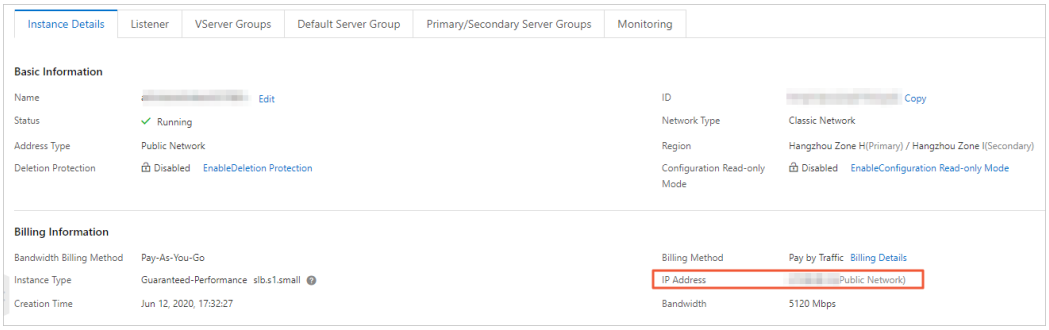
If you do not enable SSH logon when you create a cluster of Container Service for Kubernetes (ACK), you cannot connect to the cluster by using SSH or kubectl. To use SSH to connect to such a cluster after it is created, you can manually attach elastic IP addresses (EIPs) to ECS instances in the cluster, configure security group rules, and open SSH port 22.

Context

For more information about how to enable SSH logon when you create a cluster, see Bind an EIP to the Kubernetes API server when you create an ACK cluster.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the ACK console, choose **Clusters > Clusters**.
5. On the **Clusters** page, find the cluster for which you want to enable SSH logon or click **Manage** in the **Actions** column.
6. On the **details** page of the cluster, click the **Cluster Resources** tab. On the Cluster Resources tab, click the ID on the right side of **APIServer SLB**.
7. In the left-side navigation pane, choose **Instances > Instances**. On the Instances page, click the **Listener** tab. On the Listener tab, click **Add Listener**.
8. Add an SSH listener. Add an SSH listener. For more information, see Add a TCP listener.
   After the listener is created, you can use SSH to connect to your cluster through the public IP address of the SLB instance.



3.4.6. Access ACK clusters with Service Account tokens

This topic describes how to access Container Service for Kubernetes (ACK) clusters with Service Account tokens. This method is applicable to all types of ACK clusters. In this example, a managed Kubernetes cluster is used.

Context

- A managed Kubernetes cluster is created. For more information, see Create a managed Kubernetes cluster.
- The created cluster can be connected by using kubectl. For more information, see Use kubectl to connect to an ACK cluster.

Procedure

1. Run the following command to query the internal endpoint of the Kubernetes API server.

   ```
   kubectl get endpoints kubernetes
   ```

   ```
   ubuntu-mia@ubuntumia-VirtualBox:~$ kubectl get endpoints kubernetes
   NAME         ENDPOINTS           AGE
   kubernetes   [redacted]:6443     13d
   ```

2. Create a file named *kubernetes-public-service.yaml* and copy the following content to the file. Set the value of the ip field to the internal endpoint that is obtained in Step 1.

```
apiVersion: v1
kind: Service
metadata:
 name: kubernetes-public
spec:
 type: LoadBalancer
 ports:
 - name: https
   port: 443
   protocol: TCP
   targetPort: 6443
---
apiVersion: v1
kind: Endpoints
metadata:
 name: kubernetes-public
 namespace: default
subsets:
- addresses:
 - ip: <API Service address> # Replace the IP address with the internal endpoint that is obtained in Step 1.
 - name: https
   port: 6443
   protocol: TCP
```

3. Run the following command to create a LoadBalancer type Service with an external endpoint.

```
kubectl apply -f kubernetes-public-service.yaml
```

4. Run the following command to query the external Sever Load Balancer (SLB) IP address. You can find the IP address in the EXTERNAL-IP field of the result.

```
kubectl get service name
```

> **Note** You must set *name* to the value same as *name* in the *kubernetes-public-service.yaml* file created in Step 2. In this example, the value is *kubernetes-public*.

```
ubuntu-mia@ubuntumia-VirtualBox:~$ kubectl get service kubernetes-public
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
kubernetes-public   LoadBalancer                                   443:      /TCP   7d
```

5. Run the following command to query the Secret that is used to store the Service Account token. In this example, the Secret in the default *namespace* is queried.

```
kubectl get secret --namespace=namespace
```

```
ubuntu-mia@ubuntumia-VirtualBox:~$ kubectl get secret --namespace=default
NAME                          TYPE                                  DATA   AGE
aliyun-acr-credential-a       kubernetes.io/dockerconfigjson        1      13d
aliyun-acr-credential-b       kubernetes.io/dockerconfigjson        1      13d
                              kubernetes.io/service-account-token   3      13d
```

6. Run the following command to query the value of the token that is obtained in Step 5.

```
kubectl get secret -n --namespace=namespace -o jsonpath={.data.token} | base64 -d
```

> **Note** You must set *namespace* to the value same as *namespace* in Step 5.

7. Run the following command to access the managed Kubernetes cluster.

```
curl -k -H 'Authorization: Bearer token' https://service-ip
```

> **Note**
>
> - Set token to the value that is obtained in Step 6.
> - Set service-ip to external SLB IP address (the IP address in the EXTERNAL-IP field) that is obtained in Step 4.

Result

If the output is as shown in the following figure, it indicates that you are connected to the cluster.

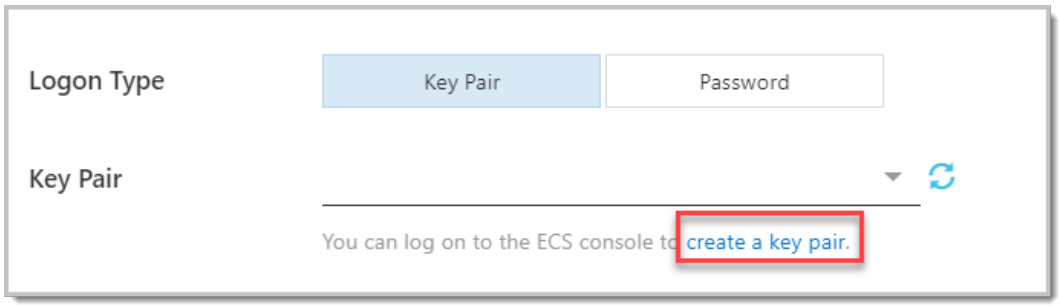
3.4.7. Use SSH key pairs to connect to an ACK cluster

This topic describes how to use SSH key pairs to log on to a cluster of Container Service for Kubernetes (ACK). This logon method secures remote access to ACK clusters.

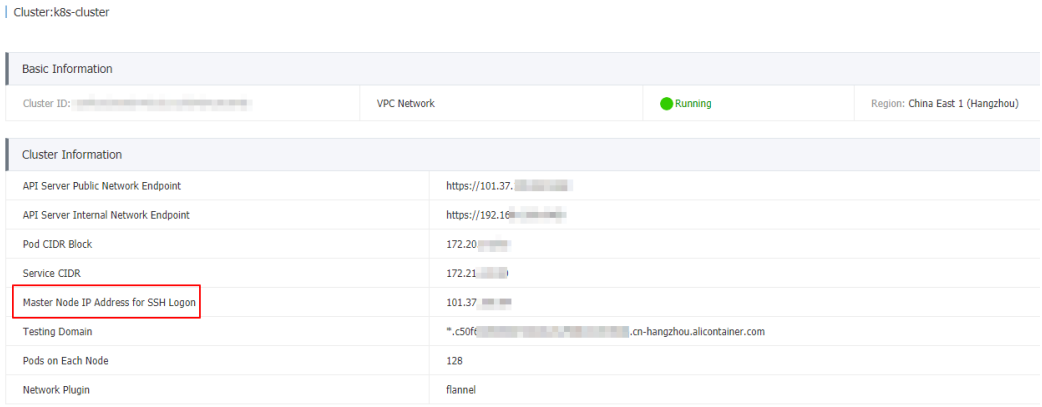
Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the left-side navigation pane of the ACK console, choose **Clusters > Clusters**.
4. In the upper-right corner of the **Clusters** page, click **Create Kubernetes Cluster**.
5. On the **Select Cluster Template** page, select a cluster type and click **create**.
6. Set the logon type to SSH key pair. Set the remaining parameters that are required to create an ACK cluster and click **Create Cluster**. For more information about the parameters, see 创建Kubernetes托管版集群.
   i. If you have created an SSH key pair in the Elastic Compute Service (ECS) console, you can select it from the Key Pair drop-down list.

ii. If no SSH key pair is available, click **create a key pair** to create one in the ECS console. For more information, see Create an SSH key pair.



7. After the cluster is created, go to the Clusters page. On the Clusters page, find the newly created cluster and click **Details** in the Actions column. On the **Basic Information** tab, you can obtain the **IP address for SSH connection** to a master node, as shown in the following figure.



8. Download the *.pem* private key file and configure SSH logon based on the operating system (Windows or Linux) of your local machine. For more information, see 创建Kubernetes托管版集群. In this example, the Linux operating system is used.
   i. Find the downloaded *.pem* private key file, for example, `/root/xxx.pem`.
   ii. Run the following command to modify the attributes of the file: `chmod 400 [path of .pem]`. Example: `chmod 400 /root/xxx.pem`.
   iii. Run the following command to use SSH to connect the cluster: `ssh -i [path of .pem] root@[master-public-ip]`. Replace master-public-ip with the IP address that you have obtained in Step 7, for example, `ssh -i /root/xxx.pem root@10.10.xx.xxx`.

3.4.8. Control internal access to the API server by using SLB

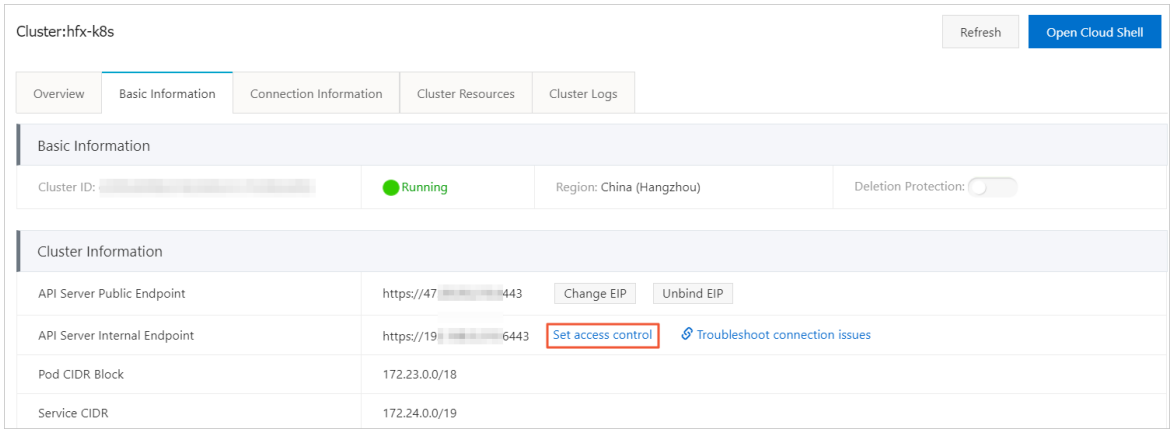
To enforce internal access control for the API server of a Container Service for Kubernetes (ACK) cluster, you can set a whitelist or blacklist for the listener that listens on Port 6443 in the Server Load Balancer (SLB) console. This topic describes how to control internal access to the API server by using SLB.

Context

SLB allows you to enforce access control by using listeners. You can configure access control when you create a listener or modify access control settings for an existing listener. For more information, see Overview.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the **Cluster Information** page, click the **Basic Information** tab.
5. On the **Basic Information** tab, find the **API Server Internal Endpoint** field in the **Cluster Information** section, and click **Set access control** next to the field.



6. You are redirected to the **Access Control Settings** panel in the SLB console. Turn on **Enable Access Control**, set Access Control Method to Whitelist or Blacklist, and then select the required access control list (ACL).

   Before you enable access control, you must create the required ACL. For more information about how to enable access control, see Enable access control.

   > **Notice**
   > If you set a whitelist, you must add the CIDR block 100.104.0.0/16 of the ACK cluster and the CIDR block of the vSwitch to the whitelist. The vSwitch is used by the master node on which the API server runs.
   > If you set a blacklist, you must not add the CIDR block 100.104.0.0/16 of the ACK cluster and the CIDR block of the vSwitch to the blacklist. The vSwitch is used by the master node on which the API server runs.

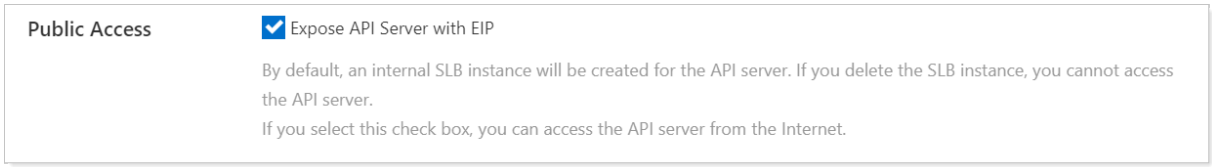
7. Click **OK**.

3.4.9. Access the Kubernetes API server over the Internet

You can use an elastic IP address (EIP) to expose the API server for a cluster of Container Service for Kubernetes (ACK). After you perform this operation, you can access the API server of the ACK cluster over the Internet. You can bind an EIP to the Kubernetes API server during or after the cluster creation. This topic describes how to bind an EIP to an API server in the preceding ways.

Bind an EIP to the Kubernetes API server when you create an ACK cluster

When you create an ACK cluster, select **Expose API Server with EIP**. For information about how to create an ACK cluster, see 创建Kubernetes托管版集群.

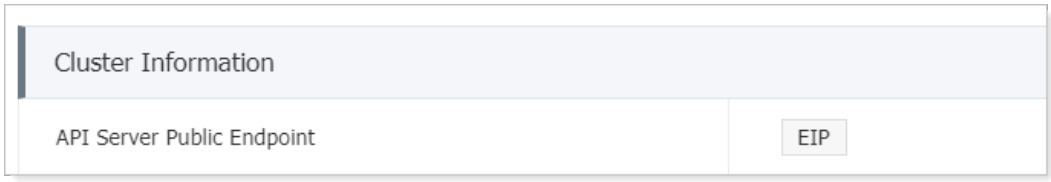


Bind an EIP to the Kubernetes API server after an ACK cluster is created

If you do not select Expose API Server with EIP when you create an ACK cluster, you can perform the following steps to bind an EIP to the Kubernetes API server of the cluster:

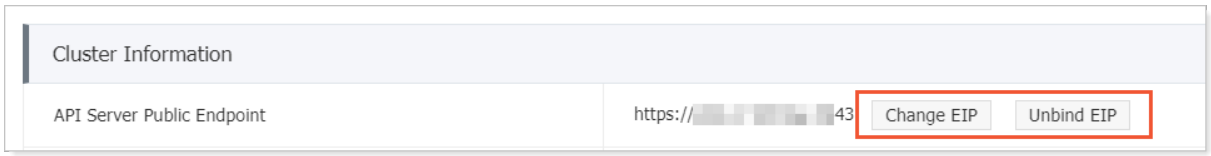
> **Notice** Only ACK Standard clusters allow you to bind an EIP to the Kubernetes API server after the cluster is created.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the left-side navigation pane of the ACK console, choose **Clusters > Clusters**.
4. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
5. On the **Clusters** page, find the cluster to which you want to bind an EIP and click **Manage** in the **Actions** column.
6. On the details page, click the **Basic Information** tab. In the **Cluster Information** section, click **Bind EIP**.



What's next

After an EIP is bound to the Kubernetes API server, you can change or unbind the EIP.



> **Notice** You can **change** or **unbind** the EIP for only ACK Standard clusters

3.5. Manage clusters

3.5.1. Cluster lifecycle

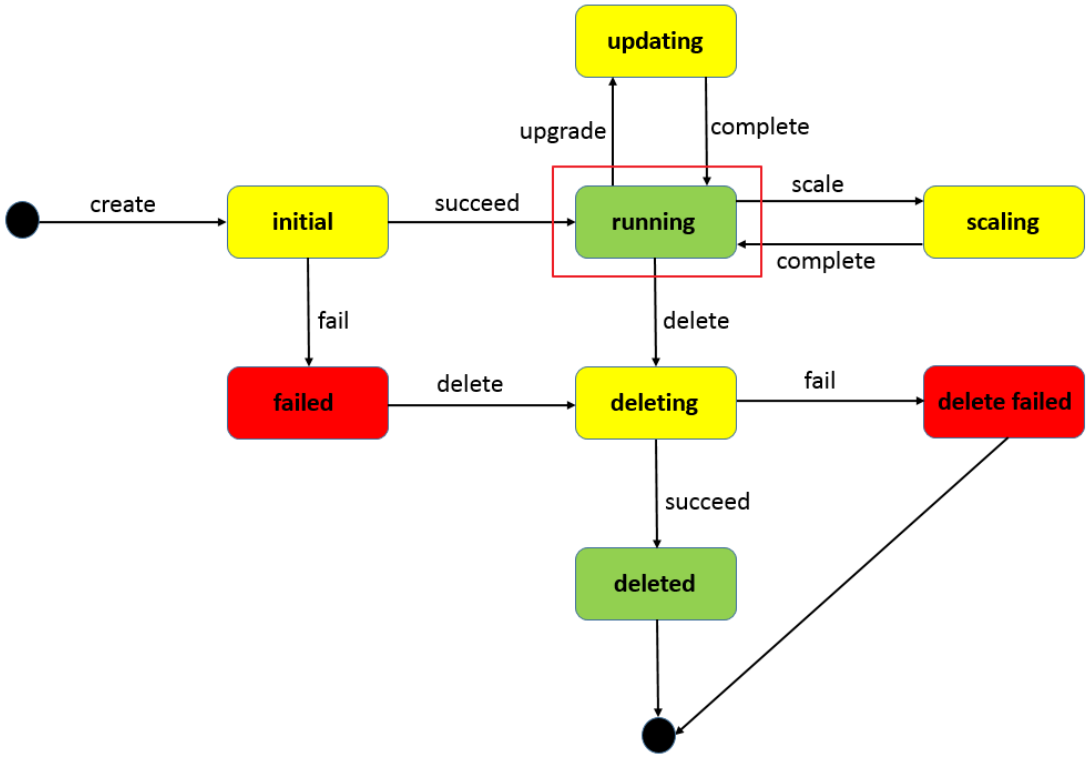
This topic describes the definitions of different cluster states and how the state of a cluster changes within a cluster lifecycle.

Cluster status

State	Description

State	Description
Initializing	The cluster is applying for corresponding cloud resources.
Creation Failed	The cluster fails to apply for corresponding cloud resources.
Running	The cluster has successfully applied for corresponding cloud resources.
Updating	The cluster is updating the metadata.
Scaling	The cluster is scaling in or out.
Removing	One or more nodes are being removed from the cluster.
Upgrading	The cluster is upgrading.
Deleting	The cluster is being deleted.
Deletion Failed	The cluster fails to be deleted.
Deleted (invisible to users)	The cluster is deleted.

Cluster status changes



3.5.2. View cluster information

Container Service for Kubernetes (ACK) provides an overview page for each cluster in the ACK console. You can view the statuses of applications and components, and check the monitoring information about the computing resources. This allows you to understand the health status of the cluster. On the overview page, you can also view the basic information, connection information, computing resources, logs, and time zone of the cluster.

Check the cluster overview

On the overview page of an ACK cluster, you can view the statuses of applications, nodes, and components in the cluster. You can also view the monitoring information about the computing resources.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. Click the **Overview** tab to go to the overview page of the cluster.
5. Select the namespace that you want to check. You can view the diagrams on the page to check the statuses of applications and components in the namespace. You can also view the monitoring charts of the computing resources in the namespace.
   - **Application Status**: This section displays diagrams that show the statuses of the Deployments, pods, and StatefulSets that are running in the selected namespace. The green color indicates that applications are running as normal. The yellow color indicates that exceptions have occured.
   - **Node Status**: This section displays a diagram that shows the statuses of the nodes in the cluster.
   - **Component Status**: In most cases, cluster components are deployed in the kube-system namespace. Core components, such as kube-scheduler, Cloud Controller Manager (CCM), and etcd, are also deployed in this namespace.
   - **Monitoring**: This section displays the monitoring charts of CPU and memory resources. The CPU utilization is measured in cores and is accurate to three decimal places. The smallest unit is millicores. A millicore is one-thousandth of a core. The memory utilization is measured in GiB and is accurate to three decimal places. For more information, see Meaning of CPU and Meaning of memory.
   - **Events**: This section displays events in the cluster. For example, warnings and errors.

View basic information

On the **Basic Information** tab, you can view the cluster ID, region, API server endpoint, and network information.

View connection information

On the **Connection Information** tab, you can obtain the kubeconfig file that is used to connect to the cluster over the Internet or the internal network. You can also use kubectl to connect to the cluster based on the content in the kubeconfig file.

View cluster resources

On the **Cluster Resources** tab, you can view the cloud resources that are used by the cluster. To check a resource in the corresponding console, click the resource ID.

> **Notice** The cluster resources are managed by ACK. We recommend that you do not delete or modify these resources. Otherwise, exceptions may occur and applications that run in the cluster may also be affected.

View cluster logs

You can view cluster logs by using the following methods:

- Method 1: View cluster logs through the **Cluster Information** menu.
  In the left-side navigation pane of the cluster details page, click Cluster Information. Click the **Cluster Logs** tab to view the cluster logs.
- Method 2: View cluster logs through the **Operations** menu.
  In the left-side navigation pane of the cluster details page, choose **Operations > Log Center**. On the Log Center page, click the **Cluster Logs** tab to view the cluster logs.

View the time zone of a cluster

You can view the time zone of a cluster and the time zone of a worker node.

- To view the time zone of a cluster, connect to the cluster by using kubectl and run the following command:

  ```
  kubectl get configmap -n kube-system ack-cluster-profile -o yaml | grep timezone
  ```

  Sample output: `timezone: Asia/Shanghai`
- To view the time zone of a worker node, connect to the worker node by using SSH and run the following command:

  ```
  ls -l /etc/localtime
  ```

  Sample output: `lrwxrwxrwx 1 root root 30 Sep 30 18:44 /etc/localtime -> /usr/share/zoneinfo/Asia/Shanghai`

3.5.3. Cluster certificates

3.5.3.1. Renew cluster certificates

This topic describes how to renew the certificates used in a cluster.
About two months before a certificate expires, a red button appears on the Clusters page in the console to remind you to renew the certificate. You will also receive an internal message or Short Message Service (SMS) notification about the expiring certificate.

You can click the button to renew the certificate. It requires about 5 to 10 minutes to complete the process based on the number of nodes in the cluster. After the certificate is renewed, its validity period is extended by five years.

Notes

- During the renewal process, the following system components will be restarted: kube-apiserver, kube-controller-manager, kube-scheduler, and kubelet. If your business logic is strongly reliant on system components such as apiserver, make sure that your services are available during the renewal before you start the process.
- We recommend that you renew certificates during off-peak hours.

Backup

Node type	Backup content

Node type	Backup content
Master	• /etc/kubernetes/ • /var/lib/kubelet/pki • /etc/systemd/system/kubelet.service.d/10-kubeadm.conf • /etc/kubeadm/ • *Critical business data* **Note** If */var/lib/kubelet/pki* is empty or you have no *critical business data*, backup is not required.
Worker	• /etc/kubernetes/ • /etc/systemd/system/kubelet.service.d/10-kubeadm.conf • /var/lib/kubelet/pki/* • *Critical business data* **Note** If */var/lib/kubelet/pki/** is empty or you have no *critical business data*, backup is not required.

Certificates

Master node

Certificate or conf filename	Path
• apiserver.crt • apiserver.key	/etc/kubernetes/pki
• apiserver-kubelet-client.crt • apiserver-kubelet-client.key	/etc/kubernetes/pki
• front-proxy-client.crt • front-proxy-client.key	/etc/kubernetes/pki
• dashboard.crt • dashboard.key	/etc/kubernetes/pki/dashboard
• kubelet.crt • kubelet.key **Note** If file *kubelet.key* does not exist, renewal is not required.	/var/lib/kubelet/pki **Note** If this path is empty, renewal is not required.
admin.conf	/etc/kubernetes

Certificate or conf filename	Path
kube.conf	/etc/kubernetes
controller-manager.conf	/etc/kubernetes
scheduler.conf	/etc/kubernetes
kubelet.conf	/etc/kubernetes
config	~/.kube/
• kubelet-client-current.pem or kubelet-client.crt\ • kubelet-client.key **Note** If the *kubelet-client.key* file does not exist, renewal is not required.	/var/lib/kubelet/pki **Note** If this path is empty, renewal is not required.

The worker node.

Certificate or conf filename	Path
• kubelet.crt • kubelet.key **Note** If the *kubelet.key* file does not exist, renewal is not required.	/var/lib/kubelet/pki **Note** If this path is empty, renewal is not required.
• kubelet-client-current.pem or kubelet-client.crt • kubelet-client.key **Note** If the *kubelet-client.key* file does not exist, renewal is not required.	/var/lib/kubelet/pki **Note** If this path is empty, renewal is not required.
kubelet.conf	/etc/kubernetes

3.5.3.2. Update the Kubernetes cluster certificates that are about to expire

This topic describes how to update the Kubernetes cluster certificates that are about to expire. You can use multiple methods to update the cluster certificates. For example, you can update the cluster certificates in the Container Service for Kubernetes console. You can run a single command to update all the certificates. You can also run different commands to separately update the certificates of the master and worker nodes.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- The Kubernetes cluster is connected based on kubectl. For more information, see Connect to a Kubernetes

cluster by using kubectl.

Run a command to update all certificates

Log on to a master node. In the command-line interface (CLI), run the following command:

```
curl http://aliacs-k8s-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/public/cert-update/renew.sh | bash
```

Response:

1. In the CLI, run the following command to view the status of the master nodes and worker nodes:

   ```
   kubectl get nodes
   ```

   ```
   [root@ ~]# kubectl get nodes
   NAME                    STATUS    ROLES     AGE       VERSION
   cn-hangzhou.            Ready     <none>    23d       v1.11.2
   cn-hangzhou.            Ready     <none>    23d       v1.11.2
   cn-hangzhou.            Ready     master    47d       v1.11.2
   cn-hangzhou.            Ready     master    47d       v1.11.2
   cn-hangzhou.            Ready     master    47d       v1.11.2
   cn-hangzhou.            Ready     <none>    47d       v1.11.2
   cn-hangzhou.            Ready     <none>    47d       v1.11.2
   [root@ ~]#
   ```

2. In the CLI, run the following command. The response shows that the SUCCESSFUL parameter value of each master node is *1*, and the SUCCESSFUL parameter value of each worker node indicates the number of worker nodes in the cluster. In this case, all certificates are updated.

   ```
   kubectl get job –nkube-system
   ```

   ```
   [root@ ~]# kubectl get job -nkube-system
   NAME                        DESIRED   SUCCESSFUL   AGE
   aliyun-cert-renew-master-1  1         1            6m
   aliyun-cert-renew-master-2  1         1            5m
   aliyun-cert-renew-master-3  1         1            5m
   aliyun-cert-renew-worker    4         4            4m
   cert-job-2                  1         1            22h
   cert-job-3                  1         1            22h
   cert-job-4                  1         1            22h
   cert-node-2                 4         4            19h
   ```

Manually update the certificates of each master node

1. Copy and paste the following code into any path to create a *job-master.yml* file:

```
apiVersion: batch/v1
kind: Job
metadata:
 name: ${jobname}
 namespace: kube-system
spec:
 backoffLimit: 0
 completions: 1
 parallelism: 1
 template:
  spec:
   activeDeadlineSeconds: 3600
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - ${hostname}
   containers:
   - command:
    - /renew/upgrade-k8s.sh
    - --role
    - master
    image: registry.cn-hangzhou.aliyuncs.com/acs/cert-rotate:v1.0.0
    imagePullPolicy: Always
    name: ${jobname}
    securityContext:
     privileged: true
    volumeMounts:
    - mountPath: /alicoud-k8s-host
     name: ${jobname}
   hostNetwork: true
   hostPID: true
   restartPolicy: Never
   schedulerName: default-scheduler
   securityContext: {}
   tolerations:
   - effect: NoSchedule
    key: node-role.kubernetes.io/master
   volumes:
   - hostPath:
     path: /
     type: Directory
    name: ${jobname}
```

2. Obtain the number of master nodes in the cluster and the hostname of each master node.
   - Method 1:
     In the CLI, run the following command:

     ```
     kubectl get nodes
     ```

```
[root@                    ~]# kubectl get nodes
NAME                        STATUS    ROLES     AGE       VERSION
cn-hangzhou.i               Ready     <none>    22d       v1.11.2
cn-hangzhou.i               Ready     <none>    22d       v1.11.2
cn-hangzhou.i               Ready     master    46d       v1.11.2
cn-hangzhou.i               Ready     master    46d       v1.11.2
cn-hangzhou.i               Ready     master    46d       v1.11.2
cn-hangzhou.i               Ready     <none>    46d       v1.11.2
cn-hangzhou.i               Ready     <none>    46d       v1.11.2
[root@                    ~]#
```

- Method 2:
  a. Log on to the Container Service for Kubernetes console.
  b. In the left-side navigation pane, click **Clusters**.
  c. On the **Clusters** page, click the name of the cluster that you want to manage, or click **Details** in the Actions column of the cluster.
  d. The Cluster Information page appears. In the left-side navigation pane, click **Nodes**. On the **Nodes** page, check the number of master nodes, and the name, IP address, and instance ID of each node.

3. In the CLI, run the following command to set the ${jobname} and ${hostname} variables in the *job-master.yml* file:

   ```
   sed 's/${jobname}/cert-job-2/g; s/${hostname}/hostname/g' job-master.yml > job-master2.yml
   ```

   where,
   - ${jobname} is the job and pod name. In this example, this variable is set to *cert-job-2*.
   - ${hostname} is the master name. In this example, *hostname* is set to a master name that is returned in Step 2.
4. In the CLI, run the following command to create a job:

   ```
   kubectl create -f job-master2.yml
   ```

5. In the CLI, run the following command to view the job status. When the SUCCESSFUL parameter value is *1*, the certificates of this master node have been updated.

   ```
   kubectl get job –nkube-system
   ```

6. Repeat Steps 3 to 5 to update the certificates of the other master nodes in the cluster.

```
[root@                    ~]# kubectl get job -nkube-system
NAME         DESIRED   SUCCESSFUL   AGE
cert-job-2   1         1            22m
cert-job-3   1         1            2m
cert-job-4   1         1            1m
[root@                    ~]#
```

Manually update worker node certificates

1. Copy and paste the following code into any path to create a *job-node.yml* file:

```
apiVersion: batch/v1
kind: Job
metadata:
 name: ${jobname}
 namespace: kube-system
spec:
 backoffLimit: 0
 completions: ${nodesize}
 parallelism: ${nodesize}
 template:
  spec:
   activeDeadlineSeconds: 3600
   affinity:
    podAntiAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
     - labelSelector:
        matchExpressions:
        - key: job-name
         operator: In
         values:
         - ${jobname}
       topologyKey: kubernetes.io/hostname
   containers:
   - command:
    - /renew/upgrade-k8s.sh
    - --role
    - node
    - --rootkey
    - ${key}
    image: registry.cn-hangzhou.aliyuncs.com/acs/cert-rotate:v1.0.0
    imagePullPolicy: Always
    name: ${jobname}
    securityContext:
     privileged: true
    volumeMounts:
    - mountPath: /alicoud-k8s-host
     name: ${jobname}
   hostNetwork: true
   hostPID: true
   restartPolicy: Never
   schedulerName: default-scheduler
   securityContext: {}
   volumes:
   - hostPath:
      path: /
      type: Directory
    name: ${jobname}
```

> **Note** If a worker node has a taint, add `tolerations` for the taint in the *job-node.yml* file. You must add the following code between `securityContext: {}` and `volumes:`. If the number of worker nodes that have taints is *n*, you must add the following code *n* times).
>
> ```
> tolerations:
> - effect: NoSchedule
>  key: ${key}
>  operator: Equal
>  value: ${value}
> ```
>
> To obtain the values of ${name} and ${value}, perform the following steps:
>
> i. Copy and paste the following code into any path to create a *taint.tml* file:
>
> ```
> {{printf "%-50s %-12s\n" "Node" "Taint"}}
> {{- range .items}}
>   {{- if $taint := (index .spec "taints") }}
>     {{- .metadata.name }}{{ "\t" }}
>     {{- range $taint }}
>       {{- .key }}={{ .value }}:{{ .effect }}{{ "\t" }}
>     {{- end }}
>     {{- "\n" }}
>   {{- end}}
> {{- end}}
> ```
>
> ii. Run the following command to view the values of ${name} and ${value} for the worker nodes that have taints:
>
> ```
> kubectl get nodes -o go-template-file="taint.tml"
> ```
>
> ```
> [root@                          ~]# kubectl get nodes -o go-template-file="taint.tml"
> Node                                                Taint
> cn-hangzhou.i-                      key1=value1:NoSchedule
> cn-hangzhou.i-                      node-role.kubernetes.io/master=<no value>:NoSchedule
> cn-hangzhou.i-                      node-role.kubernetes.io/master=<no value>:NoSchedule
> cn-hangzhou.i-                      node-role.kubernetes.io/master=<no value>:NoSchedule
> ```

2. In the CLI, run the following command to obtain the cluster CA key:

   ```
   sed '1d' /etc/kubernetes/pki/ca.key | base64 -w 0
   ```

3. In the CLI, run the following command to set the ${jobname}, ${nodesize}, and ${key} variables in the *job-node.yml* file:

   ```
   sed 's/${jobname}/cert-node-2/g; s/${nodesize}/nodesize/g; s/${key}/key/g' job-node.yml > job-node2.yml
   ```

   where,

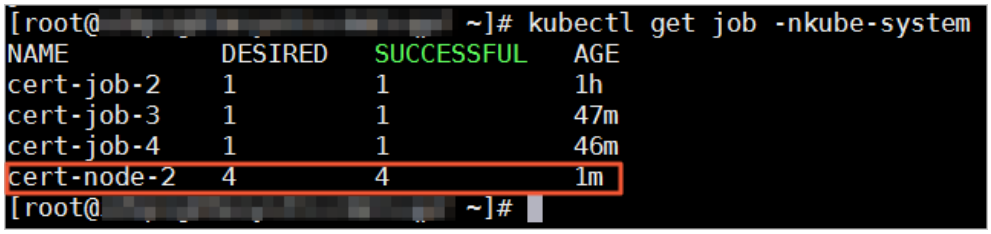
   - ${jobname} is the job and pod name. In this example, this variable is set to *cert-node-2*.
   - ${nodesize} is the number of worker nodes. For more information about how to obtain this value, see Step 1 in Manually update worker node certificates. In this example, the *nodesize* variable is replaced with the number of the worker nodes in the cluster.
   - ${key} is the cluster CA key. In this example, the *key* variable is replaced with the CA key that is returned in Step 2 of Manually update worker node certificates.

4. In the CLI, run the following command to create a job:

   ```
   kubectl create –f job-node2.yml
   ```

5. Run the following command to view the job status. When the SUCCESSFUL parameter value is equal to the number of the cluster worker nodes, all certificates have been updated.

```
kubectl get job –nkube-system
```



3.5.3.3. Update expired certificates of a Kubernetes cluster

If the certificates that are used on the nodes of a Kubernetes cluster expire, you cannot communicate with the cluster API server by using kubectl or calling API operations. The expired certificates cannot be automatically updated based on template deployment. To update the certificates, you can log on to each node and run the `docker run` command.

Update the expired certificate on each master node

1. Log on to a master node as the root user.
2. In the command-line interface (CLI), run the following command in any directory. This allows you to update the expired certificate on the master node:

```
docker run -it --privileged=true  -v /:/alicoud-k8s-host --pid host --net host \
 registry.cn-hangzhou.aliyuncs.com/acs/cert-rotate:v1.0.0 /renew/upgrade-k8s.sh --role master
```

3. Repeat the preceding steps on each cluster master node to update all the expired certificates.

Update the expired certificate on a worker node

1. Log on to a master node as the root user.
2. In the CLI, run the following command to obtain the cluster rootCA private key:

```
cat /etc/kubernetes/pki/ca.key
```

3. Run either of the following commands to obtain the cluster root private key that is encoded in the Base64 format:
   - If the cluster rootCA private key contains a blank line, run the following command:

     ```
     sed '1d' /etc/kubernetes/pki/ca.key| base64 -w 0
     ```

   - If the cluster rootCA private key does not contain a blank line, run the following command:

     ```
     cat /etc/kubernetes/pki/ca.key | base64 -w 0
     ```

4. Log on to a worker node as the root user.
5. In a directory on the worker node, run the following command to update the expired certificate on the worker node.

```
docker run -it --privileged=true  -v /:/alicoud-k8s-host --pid host --net host \
 registry.cn-hangzhou.aliyuncs.com/acs/cert-rotate:v1.0.0 /renew/upgrade-k8s.sh --role node --rootkey ${base64CAKey}
```

> **Note** ${base64CAKey} specifies the cluster root private key that is encoded in the Base64 format. The value of ${base64CAKey} is returned in Step 3.

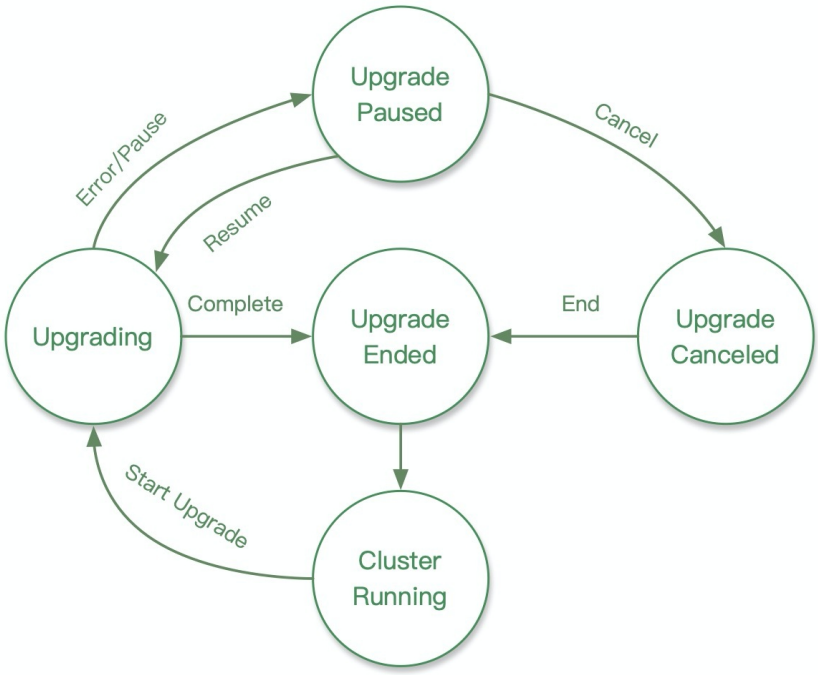
6. Repeat the preceding steps on each worker node of the cluster to update all the expired certificates.

3.5.4. Upgrade a cluster

This topic describes how to upgrade the Kubernetes version of your cluster. You can go to the Clusters page to check the Kubernetes version of your cluster and check whether the existing version can be upgraded. The cluster upgrade process involves the following phases: precheck, master node upgrade, and node upgrade. For dedicated clusters, the serial number of the master node that is being upgrade will be displayed during master node upgrade. The serial number starts from 1. During node upgrade, the information about upgraded nodes and total nodes are displayed.

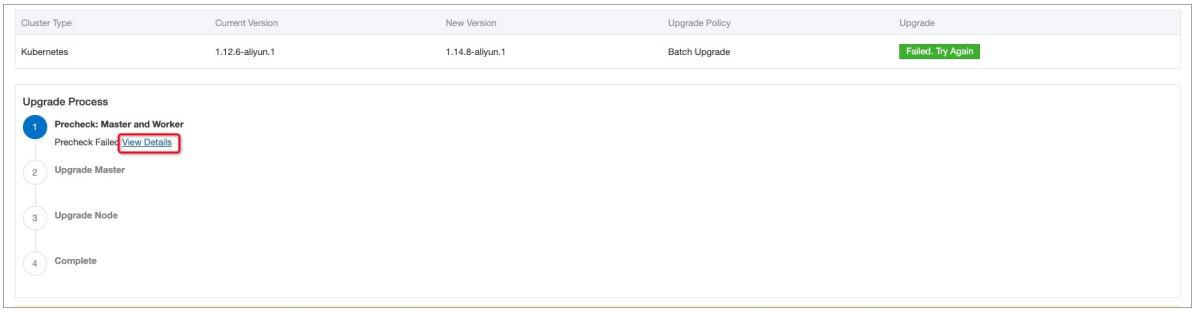
How the upgrade works

The following figure shows how the upgrade works. It also provides more information about the steps that are involved in the upgrade process.

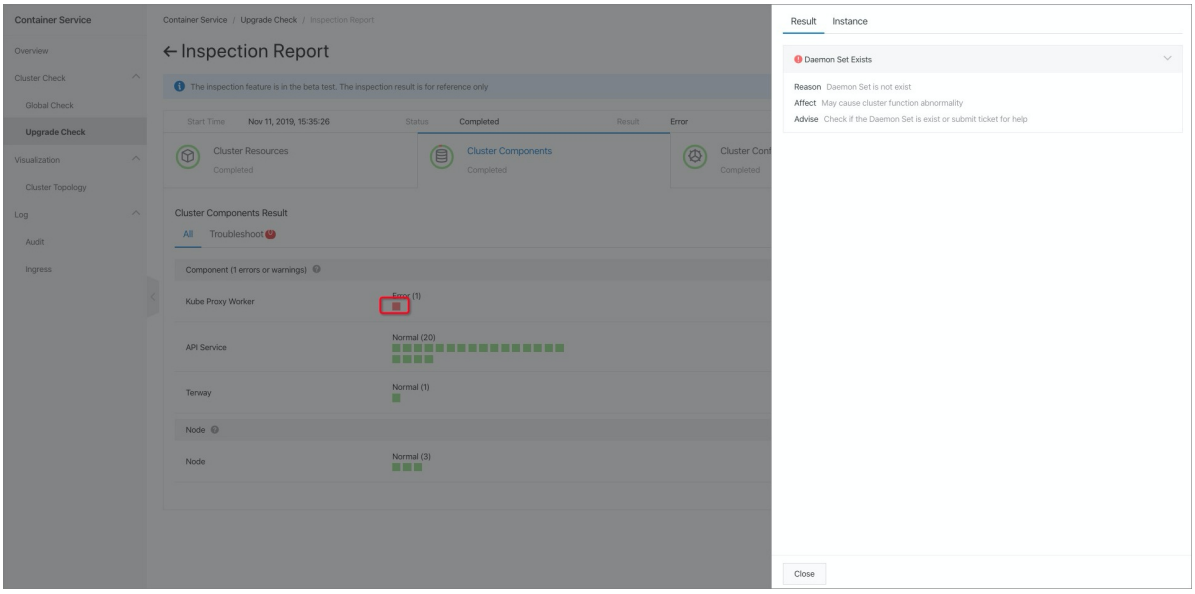


- Upgrade policy
  The upgrade policy defines how the upgrade is implemented. The default policy is batch upgrade. The batch upgrade policy is used during **node upgrade**. This allows you to upgrade multiple nodes in the cluster at a time. The policy works in the following way:
  - The first batch includes one node. In subsequent batches, the number of nodes is increased by the power of 2. If you pause the upgrade, the first batch after the pause includes one node. In subsequent batches, the number of nodes is increased by the power of 2.
  - The maximum number of nodes in each batch does not exceed 10% of the total number of nodes.
- Precheck

When you start the upgrade process, a precheck is automatically started to detect potential upgrade issues. The system performs multiple health checks for the cluster. This ensures a successful upgrade of the cluster.
If your cluster contains configuration errors or potential risks, the precheck may fail. The following figure shows a precheck failure.



Click **View Details**. You are redirected to the details page. On this page, you can check the cause of the failure.



> **Note**
> - If the precheck fails, you must fix the issue. If you require technical support, Submit a ticket.
> - A precheck is required only before the cluster upgrade. A failed precheck does not affect the running of the cluster.
> - If the cluster passes the precheck, the upgrade process automatically starts.

- Pause the upgrade
  You can pause the upgrade process.

> **Note**
> - After you pause the upgrade, the upgrade will be completed on nodes where the upgrade has already started. The upgrade will not be performed on nodes on which the upgrade has not started.
> - We recommend that you do not adjust the cluster settings when the upgrade is paused. We recommend that you resume the upgrade at your earliest convenience.

To resume the upgrade, click **Continue**. This allows you to resume the upgrade process.
If an error occurs during the upgrade, the system pauses the upgrade process. The cause of the error appears at the bottom of the page. Based on the error message, you can troubleshoot the error or Submit a ticket to seek further technical support.

- Cancel the upgrade
  After the upgrade is paused, you can click **Cancel** to cancel the upgrade.

  > **Note**
  > - After you cancel the upgrade, the upgrade will be completed on nodes where the upgrade has already started. The upgrade will not be performed on nodes on which the upgrade has not started.
  > - Nodes on which the upgrade has been completed are unaffected.

Precautions

- To upgrade a cluster, nodes in the cluster must support access over the Internet. This allows you to download additional upgrade packages.
- Failures may occur during the upgrade process. To ensure data security, we recommend that you create snapshots of volumes before you start the upgrade. For more information about how to create a snapshot of an Elastic Compute Service (ECS) instance, see Create a snapshot for a disk.
- Applications that run in the cluster are not interrupted during the upgrade. Applications that are dependent on the API server may be temporarily interrupted.
- Object Storage Service (OSS) volumes that are mounted to the Kubernetes cluster based on FlexVolume 1.11.2.5 or earlier will be remounted during the upgrade. You must recreate the pods that use OSS volumes after the upgrade is completed.
- You can modify the configurations of the cluster during the upgrade process. For example, you can create SWAP partitions. In this case, the upgrade may fail.
- You can pause the upgrade after multiple nodes are upgraded. We recommend that you do not adjust the cluster settings when the upgrade is paused. We recommend that you resume the upgrade at your earliest convenience. If the upgrade is paused for more than 15 days, the system automatically terminates the upgrade process. Then, the events and log data that have been generated during the upgrade process are deleted.
- During the upgrade process, do not modify the resources that belong to the kube-upgrade namespace unless an error has occurred.
- If an error occurs during the upgrade, the upgrade is paused. You must troubleshoot the error and delete the failed pods that belong to the kube-upgrade namespace. You can restart the upgrade after the error is fixed. If you require technical support, contact Alibaba Cloud customer service.
- After the upgrade is completed, we recommend that you upgrade the kubectl on your local worker node. Otherwise, the kubectl version may not be compatible with the API server version. In this case, the error message **invalid object doesn't have additional properties** may appear.

Before you begin

> **Note** If the cluster that you want to upgrade is not deployed in the production environment, before you start the upgrade in the production environment, we recommend that you verify that the cluster meets the upgrade requirements.

Before you upgrade a cluster, check the health status of the cluster to make sure that the cluster meets upgrade requirements.

1. Log on to the ACK console.

2. In the left-side navigation pane of the ACK console, choose **Clusters > Clusters**.
3. In the left-side navigation pane of the ACK console, click **Clusters**.
4. On the **Clusters** page, find the cluster on which you want to perform a check and choose **More > Cluster Check** in the **Actions** column.
5. In the left-side navigation pane of the **Container Service Operation Center** page, choose **Cluster Check > Upgrade Check**.
6. On the **Upgrade Check** page, click **Start**.
7. In the **Upgrade Check** pane, select the check box under **Warning** and click **Start**.
   After the upgrade check is completed, click **Details**.
   If **Cluster Resources Result** is **Normal** in the report, it indicates that the cluster passes the check and you can perform upgrade operations.

If issues are found in the cluster, you must fix the issues before you can upgrade the cluster. If you require technical support, Submit a ticket.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Clusters > Clusters**.
3. In the left-side navigation pane of the ACK console, click **Clusters**.
4. On the **Clusters** page, find the cluster that you want to upgrade and choose **More > Upgrade Cluster** in the **Actions** column.
5. Click **Upgrade**.
6. In the dialog box that appears, click **Confirm**.

   You can view the progress of the upgrade.

   > **Note**
   > - If you want to pause the upgrade, click **Pause**.
   > - After the upgrade is paused, you can click **Cancel** to cancel the upgrade process.

After the upgrade is completed, you can go to the Clusters page and check the current Kubernetes version of your edge cluster.

3.5.5. Expand an ACK cluster

This topic describes how to scale out the worker nodes in a cluster of Container Service for Kubernetes (ACK) in the ACK console.

Context

Master nodes cannot be scaled out in an ACK cluster.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to expand and choose **More > Expand** in the **Actions** column.
4. On the **Expand** page, set the scale-out parameters.

Parameter	Description
Region	This parameter is automatically set to the region where the cluster is deployed.
Container Runtime	This parameter is automatically set to **Docker 19.03.5**.
VPC	Set the network for the nodes. You can select a virtual private cloud (VPC) from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.
VSwitch	Set the vSwitches. You can select up to three vSwitches that are deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.
Billing Method	The **pay-as-you-go** and **subscription** billing methods are supported. For more information, see Billing method overview. If you select the **subscription** billing method, ◦ the **Duration** parameter is required. You can select 1, 2, 3, or 6 months. If you require a longer duration, you can select 1 to 5 years. ◦ Specify whether to enable **Auto Renewal**.
Instance Type	You can select one or more instance types. For more information, see Instance families.
Selected Types	The selected instance types are displayed.
System Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. Note ◦ You can select **Enable Backup** to back up disk data. ◦ If you select **enhanced SSD** as the system disk type, you can set a custom **performance level** for the system disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.
Mount Data Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. You can enable **disk encryption** and **disk backup** when you mount a data disk.
Nodes to Add	The number of nodes to be added to the cluster. You can add up to 100 nodes to a cluster under the current account. You can add up to 500 nodes at a time.

Parameter	Description
Logon Type	○ Key pair logon ■ **Key Pair**: Select an SSH key pair from the drop-down list. ■ **create a key pair**: Create an SSH key pair if none is available. For more information about how to create an SSH key pair, see Create an SSH key pair. After the key pair is created, set it as the credential that is used to log on to the cluster. ○ Password logon ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.
ECS Label	You can add labels to the ECS instances.
Node Label	You can add labels to the nodes of the cluster.
Taints	You can add taints to all worker nodes in the cluster.
Custom Image	You can select a custom image for the nodes to be added. After you select a custom image, the nodes to be added are deployed based on the image. For more information about how to create a custom image, see Create a Kubernetes cluster by using a custom image. **Note** ○ Only custom images based on CentOS 7.x and Alibaba Cloud Linux 2.x are supported. Before you use a custom image to deploy the nodes, you must add the nodes to the cluster in **manual** mode. For more information about how to add nodes to a cluster in Manual mode, see Manually add ECS instances. ○ This feature is available to only users in the whitelist. If you are not in the whitelist, submit a ticket.
RDS whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the ACK cluster to the RDS whitelist. **Note** To enable an RDS instance to access an ACK cluster, you must deploy the RDS instance in the same VPC as the ACK cluster.
User Data	For more information, see Overview of ECS instance user data.

5. Click **Confirm**.
   On the **Node Pools** page, if the **Status** column shows **Scaling** for the node pool, it indicates that the node pool is being scaled out. After the node pool is scaled out, the **Status** column shows **Active** for the node pool.

What's next

Click **Details** in the Actions column. On the **Nodes** tab, you can view information about the nodes that are added to the node pool.

3.5.6. Install the metrics-server component

This topic describes how to install the metrics-server component. In this case, you do not need to upgrade your Kubernetes cluster.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a dedicated Kubernetes cluster.
- Kubernetes version 1.12.6 or earlier is used for the cluster.

To install the metrics-server component, perform the following steps: change the metric collector, switch the monitoring link, and update component settings.

Change the metric collector

1. Create a *metrics-server.yaml* file and copy the following content to the file. In the command-line interface (CLI), enter `kubectl apply -f metrics-server.yaml` to change the metric collector from Heapster to metrics-server.

```
apiVersion: v1
kind: ServiceAccount
metadata:
 name: admin
 namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
 name: admin
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: cluster-admin
subjects:
- kind: ServiceAccount
 name: admin
 namespace: kube-system
---
apiVersion: v1
kind: Service
metadata:
 labels:
  task: monitoring
  # For use as a Cluster add-on (https://github.com/kubernetes/kubernetes/tree/master/cluster/addons)
  # If you are NOT using this as an addon, you should comment out this line.
  kubernetes.io/cluster-service: 'true'
  kubernetes.io/name: metrics-server
 name: heapster
 namespace: kube-system
spec:
 ports:
 - port: 80
  targetPort: 8082
 selector:
  k8s-app: metrics-server
---
apiVersion: v1
```

```
kind: Service
metadata:
 name: metrics-server
 namespace: kube-system
 labels:
  kubernetes.io/name: metrics-server
spec:
 selector:
  k8s-app: metrics-server
 ports:
 - port: 443
  protocol: TCP
  targetPort: 443
---
apiVersion: apiregistration.k8s.io/v1beta1
kind: APIService
metadata:
 name: v1beta1.metrics.k8s.io
spec:
 service:
  name: metrics-server
  namespace: kube-system
 group: metrics.k8s.io
 version: v1beta1
 insecureSkipTLSVerify: true
 groupPriorityMinimum: 100
 versionPriority: 100
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: metrics-server
 namespace: kube-system
 labels:
  k8s-app: metrics-server
spec:
 selector:
  matchLabels:
   k8s-app: metrics-server
 template:
  metadata:
   name: metrics-server
   labels:
    k8s-app: metrics-server
  spec:
   serviceAccountName: admin
   containers:
   - name: metrics-server
    image: registry. ##REGION##.aliyuncs.com/acs/metrics-server:v0.2.1-9dd9511-aliyun
    imagePullPolicy: Always
    command:
    - /metrics-server
    - '--source=kubernetes:https://kubernetes.default'
    - '--sink=socket:tcp://monitor.csk. ##REGION##.aliyuncs.com:8093? clusterId=##CLUSTER_ID##
&public=true'
```

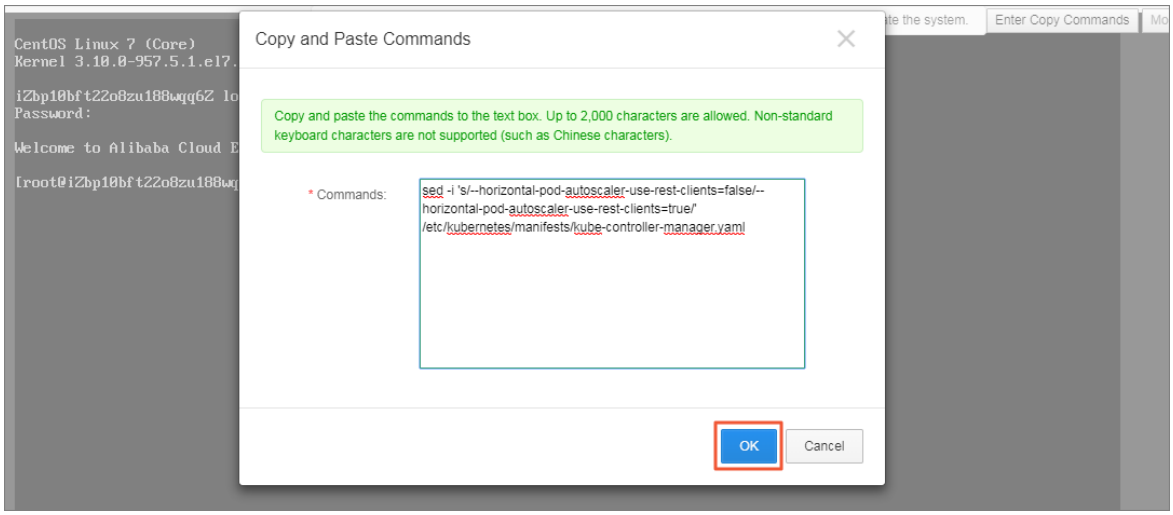
> **Note** Replace `##REGION##` and `##CLUSTER_ID##` respectively with the region (such as cn-hangzhou) and ID of the selected cluster.

Switch the monitoring link

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4.
5. Find the master nodes in the cluster. Click the ID of a master node to go to the Instance Details page.
6. In the **Basic Information** section, click **Connect**.

   The Enter VNC Password dialog box appears. Enter the VNC password and click **OK**. After you log on to the instance, run the following command:

   ```
   sed -i 's/--horizontal-pod-autoscaler-use-rest-clients=false/--horizontal-pod-autoscaler-use-rest-clients=true/' /etc/kubernetes/manifests/kube-controller-manager.yaml
   ```

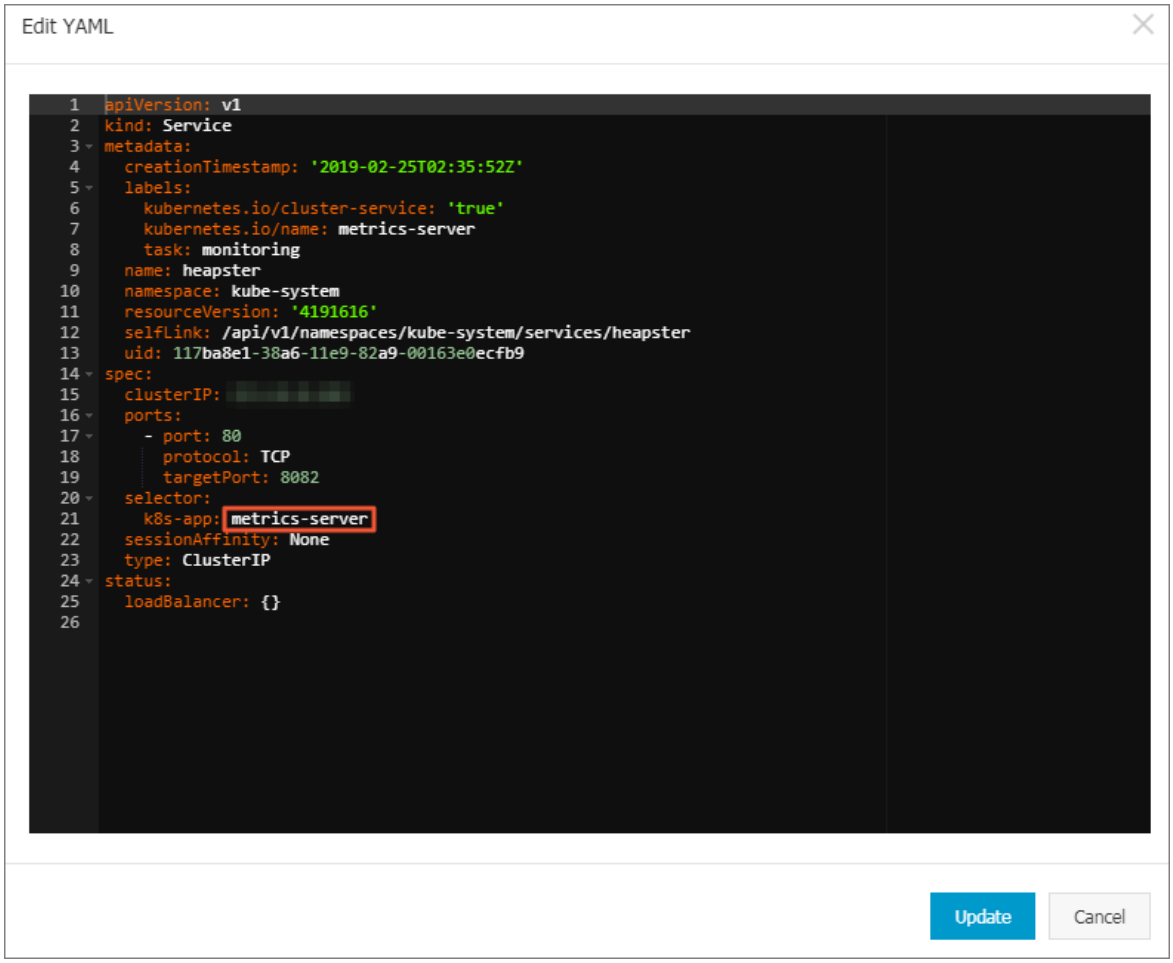
   

7. Repeat the preceding steps to switch the monitoring link for the other master nodes.

After the command is executed, the kubelet updates the controller manager.

Update component settings

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Network > Services**.
5. Select the **kube-system** namespace. Find the **heapster** service and click **View in YAML** in the **Actions** column of the service.
6. In the dialog box that appears, set the value of the k8s-app field to metrics-server. Click **Update**.



7. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
8. On the **Deployments** tab, select the **kube-system** namespace.
9. Select Heapster-related components such as heapster and monitoring-influxdb. In the **Actions** column, choose **More > Delete**.
10. In the dialog box that appears, click **OK**.

> **Note** To delete the monitoring-influxdb component, in the **Delete monitoring-influxdb** dialog box, select **Delete the associated monitoring-influxdb service** and click **OK**.

11. Check the link status.

    It requires about three minutes to initialize the link.
    In the left-side navigation pane, choose **Workload > Pods**. On the **Pods** page, if the CPU and memory usage are normal, it indicates that the link is switched.

> **Note** If the CPU and memory usage of all components are zero, it indicates that an error has occurred.

3.5.7. Manage system components

This topic describes how to upgrade, install, and uninstall system components of a Container Service for Kubernetes (ACK) cluster.

Prerequisites

Create a dedicated Kubernetes cluster

Context

In most cases, the Kubernetes version of your cluster is up-to-date. However, you may still need to separately upgrade specific system components. This topic describes how to upgrade system components.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and choose **More > Manage System Components** in the **Actions** column.
4. On the **Add-ons** page, you can perform one of the following operations:
   - Find the component that you want to install and click **Install**.
   - Find the component that you want to uninstall and click **Uninstall**.
   - Find the component that you want to upgrade and click **Upgrade**.

3.5.8. Delete an ACK cluster

This topic describes how to delete an ACK cluster in the Container Service for Kubernetes (ACK) console.

> **Notice** After you delete an ACK cluster, the nodes that you manually added to the cluster are not automatically released.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to delete and choose **More > Delete** in the Actions column.
4. In the **Delete Cluster** dialog box, verify that the cluster is the one that you want to delete and select the resources that you want to retain. Then, select **I understand the above information and want to delete the specified cluster**, and click **OK**.

   The following figure shows an example of the Delete Cluster dialog box. The actual information displayed in the dialog box is based on the cluster that you want to delete.

FAQ

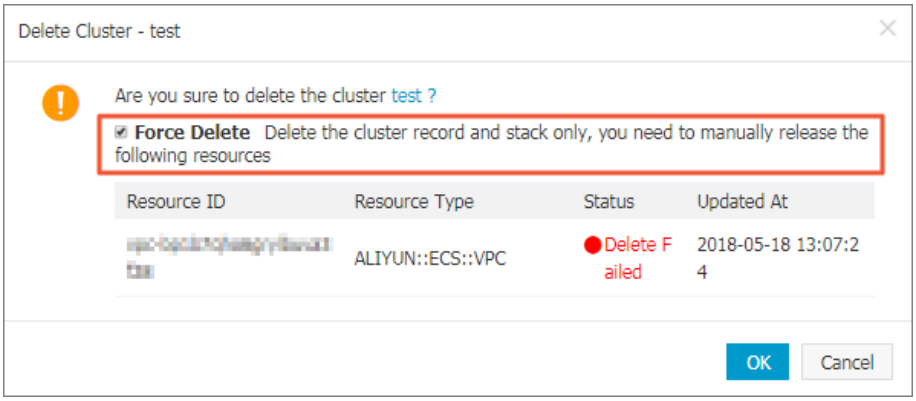
Why do I fail to delete an ACK cluster?

Resource Orchestration Service (ROS) does not have permissions to delete resources that are manually added to an ROS stack. For example, ROS cannot release a virtual private cloud (VPC) that contains a manually added vSwitch. Consequently, the cluster deployed in the VPC cannot be deleted.

ACK allows you to forcibly delete clusters. If your first attempt to delete a cluster fails, you can forcibly delete the cluster and the ROS stack. However, when you forcibly delete a cluster, the resources that are created by ROS and the manually added resources are not released. You must manually release these resources.

When a cluster fails to be deleted, the following information is displayed, as shown in the figure.

1. Find the cluster that fails to be deleted, choose **More > Delete** in the Actions column.
2. In the dialog box that appears, you can view the resources that failed to be deleted. Click **OK** to delete the cluster and ROS stack.



> **Note** If you select **Retain Resources** in the Delete Cluster dialog box, the resources that you select are not released. You must manually release these resources. For more information about how to locate resources that cannot be released, see Failed to delete Kubernetes clusters: ROS stack cannot be deleted.

3.6. FAQ about cluster management

This topic provides answers to some frequently asked questions about cluster management.

- Can I add nodes that support Intel Software Guard Extensions (Intel SGX) to an existing cluster?
- Are ACK clusters that run Alibaba Cloud Linux 2 compatible with container images of CentOS?
- How do I troubleshoot cluster creation failures?
- How do I troubleshoot cluster scale-out errors?
- How do I troubleshoot cluster deletion failures?
- Timeout errors in cluster management by using Cloud Shell

Can I add nodes that support Intel Software Guard Extensions (Intel SGX) to an existing cluster?

To add nodes that support Intel SGX to an existing cluster, the cluster must meet the following conditions:

- The Kubernetes version is 1.14.0 or later.
- The network plug-in is **Flannel**.
- The operating system is set to AliyunLinux 2.xxxx when you create the cluster. Do not select custom images when you add nodes to the cluster.

Log on to the Container Service for Kubernetes (ACK) console. On the Clusters page, find a cluster that meets the preceding requirements and choose **More > Manage System Components** in the **Actions** column. On the Add-ons page, you can install Intel SGX Architectural Enclave Service Manager (Intel SGX AESM) and sgx-device-plugin.

Are ACK clusters that run Alibaba Cloud Linux 2 compatible with container images of CentOS?

Yes. ACK clusters that run Alibaba Cloud Linux 2 are fully compatible with container images of CentOS. For more information, see Use Alibaba Cloud Linux 2.

4.Professional Kubernetes clusters

4.1. Introduction to professional managed Kubernetes clusters

Professional managed Kubernetes clusters are developed based on standard managed Kubernetes clusters. Professional managed Kubernetes clusters are covered by the service level agreement (SLA) that supports compensation clauses. This type of cluster is suitable for enterprise users that require higher stability and security for large-scale production environments.
Professional managed Kubernetes clusters offer all benefits of standard managed Kubernetes clusters. For example, master nodes are highly available and managed by Container Service for Kubernetes (ACK). In addition, professional managed Kubernetes clusters provide higher reliability, security, and schedulability and are covered by SLA terms for compensation. Professional managed Kubernetes clusters are suitable for enterprises that require higher security and stability for large-scale workloads in production environments.

Scenarios

- Internet enterprises. These enterprises deploy their business on a large scale and require business management with high stability, security, and observability.
- Big data computing enterprises. These enterprises deploy large-scale data computing services, high-performance data processing services, and other services with high elasticity. These services require clusters with high stability, high performance, and efficient computing capabilities.
- International enterprises that run their business in China. These enterprises prioritize security and services that provide SLAs with compensation clauses.
- Financial enterprises. These enterprises require SLAs with compensation clauses.

Features

- Managed master nodes with high reliability: The management of large-scale clusters is supported. etcd is a reliable store for disaster recovery and data restoration. etcd uses cold backups and hot backups to ensure data availability for professional managed Kubernetes clusters. Key metrics are collected for you to gain insights from control components. This allows you to detect potential risks.
- Clusters with higher security: By default, etcd uses encrypted disks in the control plane. In the data plane, kms-plugin is installed to encrypt Kubernetes Secrets. ACK provides security management for this type of cluster. The advanced security management feature allows you to inspect containers in the running state and enable auto repairing.
- More intelligent pod scheduling: kube-scheduler is integrated to provide better pod scheduling capabilities. This allows you to schedule pods in bulk, set multiple scheduling algorithms, and schedule pods to NPU-accelerated nodes. This also enhances the pod scheduling capability in scenarios where large-scale data computing or high-performance data processing is required.
- SLA guarantees: Professional managed Kubernetes clusters are covered by the SLA that supports compensation clauses. A level of 99.95% uptime is guaranteed for the cluster API server.

Pricing

For more information about the pricing of professional managed Kubernetes clusters, see Billing.

Comparison

The following table compares professional managed Kubernetes clusters with standard managed Kubernetes clusters.

Category	Feature	Professional managed Kubernetes Cluster	Standard managed Kubernetes cluster
Cluster size	N/A	Up to 1,000 nodes by default. You can Navigate to the Quota Center page to submit a ticket to increase the quota to 5,000.	Up to 100 nodes for each new cluster. A maximum of 1,000 nodes can be deployed in an existing standard managed Kubernetes cluster. Existing standard managed Kubernetes clusters can be upgraded to professional managed Kubernetes clusters.
SLA	N/A	99.95% (supports compensation).	99.90% (does not support compensation).
API Server	Custom parameter settings	✓	✗
	Availability monitoring	✓	✗
etcd	High-frequency cold backups, high-frequency hot backups, and geo-disaster recovery	✓	✗
	Observability metrics	✓	✗
Kube-scheduler	Gang scheduling	✓	✗
	Topology-aware CPU scheduling	✓	✗
	Topology-aware GPU scheduling	✓	✗
	cGPU Professional Edition	✓	✗
Security management	The advanced security management feature that supports data encryption. For more information, see Use KMS to encrypt Kubernetes secrets at rest in the etcd.	✓	✗

Category	Feature	Professional managed Kubernetes Cluster	Standard managed Kubernetes cluster
Managed node pool	Managed node pool	✓	✗

> **Note**
> Dedicated Kubernetes clusters support only cGPU Basic Edition. For more information, see Overview.

4.2. Create a professional managed Kubernetes cluster

Professional managed Kubernetes clusters offer higher reliability, stability, security, and schedulability than standard managed Kubernetes clusters in large-scale production environments for enterprise users. In addition, professional managed Kubernetes clusters are covered by the service level agreement (SLA) that supports compensation clauses. This topic describes how to create a professional managed Kubernetes cluster in the Container Service for Kubernetes (ACK) console.

Prerequisites

Resource Access Management (RAM) is activated in the RAM console. Auto Scaling (ESS) is activated in the ESS console.

> **Note**
> When you create a Container Service for Kubernetes (ACK) cluster, take note of the following limits:
>
> - Server Load Balancer (SLB) instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
> - ACK clusters support only VPCs.
> - By default, each account has specific quotas on cloud resources that can be created. You cannot create clusters if the quota is reached. Make sure that you have sufficient quotas before you create a cluster.
>   - For more information about the maximum numbers of clusters and nodes that can be created with each account, see Limits.
>
>   > **Note**
>
>   - By default, you can create up to 100 security groups with each account.
>   - By default, you can create up to 60 pay-as-you-go SLB instances with each account.
>   - By default, you can create up to 20 elastic IP addresses (EIPs) with each account.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Cluster Template**.
4. In the **Select Cluster Template** dialog box, select **Professional Managed Kubernetes Cluster** and click **Create**.

5. On the **Managed Kubernetes** tab, configure the cluster.
   i. Configure basic settings of the cluster.

Parameter	Description
Cluster Name	Enter a name for the ACK cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
Cluster Specification	Select a cluster type. You can select **Standard edition** or **Professional**. Select **Professional** to create a professional managed Kubernetes cluster.
Region	Select a region to deploy the cluster.
All Resources	Move the pointer over **All Resources** at the top of the page and select the resource group that you want to use. After you select a resource group, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource group. When you create a cluster, only the VPCs and vSwitches that belong to the selected resource group are displayed in the console. 
Kubernetes Version	The Kubernetes versions that are supported by ACK.
Container Runtime	The **containerd**, **Docker**, and **Sandboxed-Container** runtimes are supported. For more information, see Comparison of Docker, containerd, and Sandboxed-Container.
VPC	Select a VPC to deploy the cluster. Standard VPCs and shared VPCs are supported. ■ Shared VPC: The owner of a VPC (resource owner) can share the vSwitches in the VPC with other accounts in the same organization. ■ Standard VPC: The owner of a VPC (resource owner) cannot share the vSwitches in the VPC with other accounts. **Note** ACK clusters support only VPCs. You can select a VPC from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.
VSwitch	Select vSwitches.

Parameter	Description
Network Plug-in	Select a network plug-in. Flannel and Terway are supported. For more information, see Terway and Flannel. ■ Flannel: a simple and stable Container Network Interface (CNI) plug-in that is developed by open source Kubernetes. Flannel provides a few simple features. However, it does not support standard Kubernetes network policies. ■ Terway: a network plug-in that is developed by ACK. Terway allows you to assign elastic network interfaces (ENIs) of Alibaba Cloud to containers. It also allows you to customize Kubernetes network policies to regulate how containers communicate with each other and implement bandwidth throttling on individual containers. **Note** ■ The number of pods that can be deployed on a node depends on the number of ENIs that are attached to the node and the maximum number of secondary IP addresses that are provided by these ENIs. ■ If you select a shared VPC for an ACK cluster, you must select Terway as the network plug-in. ■ If you select **Terway**, an ENI is shared among multiple pods. A secondary IP address of the ENI is assigned to each pod.
IP Addresses per Node	If you select **Flannel** as the network plug-in, you must set **IP Addresses per Node**. **Note** ■ **IP Addresses per Node** specifies the maximum number of IP addresses that can be assigned to each node. We recommend that you use the default value. ■ After you select the VPC and specify the number of IP addresses per node, recommended values are automatically generated for Pod CIDR block and Service CIDR block. The system also provides the maximum number of nodes that can be deployed in the cluster and the maximum number of pods that can be deployed on each node. You can modify the values based on your business requirements.
Pod CIDR Block	If you select **Flannel** as the network plug-in, you must set **Pod CIDR Block**. The CIDR block specified by **Pod CIDR Block** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.

Parameter	Description
Terway Mode	If you set **Network Plug-in** to **Terway**, you must set **Terway Mode**. ■ Select or clear **Assign One ENI to Each Pod**. ■ If you select the check box, an ENI is assigned to each pod. ■ If you clear the check box, an ENI is shared among multiple pods. A secondary IP address that is provided by the ENI is assigned to each pod. **Note** To select the **Assign One ENI to Each Pod** check box, you must submit a ticket to apply to be added to a whitelist. ■ Select or clear **IPVLAN**. ■ This option is available only when you clear Assign One ENI to Each Pod. ■ If you select IPVLAN, IPVLAN and extended Berkeley Packet Filter (eBPF) are used for network virtualization when an ENI is shared among multiple pods. This improves network performance. Only the Alibaba Cloud Linux 2 operating system is supported. ■ If you clear IPVLAN, policy-based routes are used for network virtualization when an ENI is shared among multiple pods. The CentOS 7 and Alibaba Cloud Linux 2 operating systems are supported. This is the default setting. For more information about the IPVLAN feature in Terway mode, see Terway IPvlan. ■ Select or clear **Support for NetworkPolicy**. ■ The NetworkPolicy feature is available only when you clear Assign One ENI to Each Pod. By default, Assign One ENI to Each Pod is unselected. ■ If you select Support for NetworkPolicy, you can use Kubernetes network policies to control the communication among pods. ■ If you clear Support for NetworkPolicy, you cannot use Kubernetes network policies to control the communication among pods. This prevents Kubernetes network policies from overloading the Kubernetes API server.
Service CIDR	Set **Service CIDR**. The CIDR block specified by **Service CIDR** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Configure SNAT	By default, an ACK cluster cannot access the Internet. If the VPC that you select for the ACK cluster cannot access the Internet, you can select **Configure SNAT for VPC**. This way, ACK will create a NAT gateway and configure Source Network Address Translation (SNAT) rules to enable Internet access for the VPC.

Parameter	Description
Access to API Server	By default, an internal-facing Server Load Balancer (SLB) instance is created for the cluster API server. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. **Notice** If you delete the SLB instance, you cannot access the cluster API server. Select or clear **Expose API Server with EIP**. The ACK API server provides multiple HTTP-based RESTful APIs, which can be used to create, delete, modify, query, and monitor resources, such as pods and Services. ■ If you select this check box, an elastic IP address (EIP) is created and associated with an Internet-facing SLB instance. Port 6443 used by the API server is opened on master nodes. You can connect to and manage the ACK cluster by using kubeconfig over the Internet. ■ If you clear this check box, no EIP is created. You can connect to and manage the ACK cluster by using kubeconfig only within the VPC.
RDS Whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the cluster to the RDS whitelist. **Note** To enable an RDS instance to access the cluster, you must deploy the RDS instance in the VPC where the cluster is deployed.
Security Group	You can select **Create Basic Security Group**, **Create Advanced Security Group**, or **Select Existing Security Group**. For more information, see Overview. **Note** To select **Select Existing Security Group**, Submit a ticket to apply to be added to a whitelist.

ii. Configure advanced settings of the cluster.

Parameter	Description
Time Zone	Select a time zone for the ACK cluster. By default, the time zone configured for your browser is selected.

Parameter	Description
Kube-proxy Mode	iptables and IPVS are supported. ■ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the ACK cluster. This mode is suitable for ACK clusters that manage a small number of Services. ■ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for ACK clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.
Labels	Add labels to the cluster. Enter a key and a value, and click **Add**. **Note** ■ Key is required. *Value* is optional. ■ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ■ *Values* are not case-sensitive. A value can be empty and can contain up to 128 characters in length. It cannot be http:// or https://. ■ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the others that use the same key. ■ If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Cluster Domain	Set the domain name of the cluster. **Note** The default domain name is **cluster.local**. You can enter a custom domain name. A domain name consists of two parts. Each part must be 1 to 63 characters in length and can contain only letters and digits. You cannot leave these parts empty.
Custom Certificate SANs	You can enter custom subject alternative names (SANs) for the API server certificate of the cluster to accept requests from specified IP addresses or domain names. For more information, see Customize the SAN of the API server certificate for a managed Kubernetes cluster.

Parameter	Description
Service Account Token Volume Projection	**Service account token volume projection** reduces security risks when pods use service accounts to access the API server. This feature enables kubelet to request and store the token on behalf of the pod. This feature also allows you to configure token properties, such as the audience and validity duration. For more information, see Enable service account token volume projection.
Secret Encryption	If you select **Select Key**, you can use a key that is created in the Key Management Service (KMS) console to encrypt Kubernetes Secrets. For more information, see Use KMS to encrypt Kubernetes secrets at rest in the etcd.
Deletion Protection	

6. Click **Next: Worker Configurations** to configure worker nodes.
   i. Set **Worker Instance**.
   - If you select **Create Instance**, you must set the parameters as described in the following table.

Parameter	Description
Billing Method	The **pay-as-you-go** and **subscription** billing methods are supported. If you select the **subscription** billing method, you must set the following parameters: ■ **Duration**: You can select 1, 2, 3, or 6 months. If you require a longer duration, you can select 1 to 5 years. ■ **Auto Renewal**: Specify whether to enable auto-renewal.
Instance Type	You can select multiple instance types. For more information, see Instance families.
Selected Types	The selected instance types are displayed.
Quantity	Specify the number of worker nodes (ECS instances) to be created. **Note** **Quantity** can be set to 0. A professional managed Kubernetes cluster cannot contain worker nodes.

Parameter	Description
System Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. **Note** ■ You can select **Enable Backup** to back up disk data. ■ If you select **enhanced SSD** as the system disk type, you can set a custom **performance level** for the system disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.
Mount Data Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. You can enable **disk encryption** and **disk backup** when you mount a data disk.
Operating System	ACK supports the following node operating systems: ■ Alibaba Cloud Linux 2. This is the default operating system. If you select Alibaba Cloud Linux 2, you can configure security reinforcement for the operating system: ■ Disable: disables security reinforcement for Alibaba Cloud Linux 2.x. ■ CIS Reinforcement: enables security reinforcement for Alibaba Cloud Linux 2.x. For more information about CIS reinforcement, see CIS reinforcement. ■ CentOS 7.x **Note** CentOS 8.x and later are not supported.
Logon Type	■ Key pair logon ■ **Key Pair**: Select an SSH key pair from the drop-down list. ■ **create a key pair**: Create an SSH key pair if none is available. For more information about how to create an SSH key pair, see Create an SSH key pair. After the key pair is created, set it as the credential that is used to log on to the cluster. ■ Password logon ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.

- If you select **Add Existing Instance**, you must select Elastic Compute Service (ECS) instances that are deployed in the region where the cluster is deployed. Then, set **Operating System**, **Logon Type**, and **Key Pair** based on the preceding settings.

ii. Configure advanced settings of the worker nodes.

Parameter	Description
Node Protection	Specify whether to enable node protection. **Note** By default, this check box is selected. Node protection prevents nodes from being accidentally deleted in the console or by calling the API. This prevents user errors.
User Data	For more information, see Overview of ECS instance user data.
Custom Image	You can select a custom image for your ECS nodes. After you select a custom image, all nodes in the cluster are deployed by using this image. For more information about how to create a custom image, see Create a Kubernetes cluster by using a custom image. **Note** ■ Only custom images based on CentOS 7.x and Alibaba Cloud Linux 2.x are supported. ■ To use this feature, submit a ticket to apply to be added to a whitelist.
Custom Node Name	Specify whether to use a **custom node name**. A node name consists of a prefix, an IP substring, and a suffix. ■ Both the prefix and suffix can contain one or more parts that are separated by periods (.).These parts can contain lowercase letters, digits, and hyphens (-), and must start and end with a lowercase letter or digit. ■ The IP substring length specifies the number of digits to be truncated from the end of the returned node IP address. Valid values: 5 to 12. For example, if the node IP address is 192.1xx.x.xx, the prefix is aliyun.com, the IP substring length is 5, and the suffix is test, the node name will be aliyun.com00055test.
CPU Policy	Set the CPU policy. ■ none: This policy indicates that the default CPU affinity is used. This is the default policy. ■ static: This policy allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Taints	Add taints to the worker nodes in the ACK cluster.

7. Click **Next: Component Configurations** to configure components.

Parameter	Description

Parameter	Description
Ingress	Specify whether to install Ingress controllers. By default, **Install Ingress Controllers** is selected. For more information, see Ingress高级用法.
Volume Plug-in	Select a volume plug-in. FlexVolume and CSI are supported. An ACK cluster can be automatically bound to Alibaba Cloud disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets that are mounted to pods in the cluster. For more information, see Storage management-FlexVolume and Storage management-CSI.
Monitoring Agents	Specify whether to install the CloudMonitor agent. By default, **Install CloudMonitor Agent on ECS Instance** and **Enable Prometheus Monitoring** are selected. After the CloudMonitor agent is installed on ECS nodes, you can view monitoring data about the nodes in the CloudMonitor console.
Log Service	Specify whether to enable Log Service. You can select an existing Log Service project or create one. By default, **Enable Log Service** is selected. When you create an application, you can enable Log Service through a few steps. For more information, see Collect log files from containers by using Log Service. By default, **Install node-problem-detector and Create Event Center** is selected. You can also specify whether to **create Ingress dashboards** in the Log Service console.
Workflow Engine	Specify whether to enable Alibaba Cloud Genomics Service (AGS). **Note** To use this feature, submit a ticket to apply to be added to a whitelist. ○ If you select this check box, the system automatically installs the AGS workflow plug-in when the system creates the cluster. ○ If you clear this check box, you must manually install the AGS workflow plug-in. For more information, see Introduction to AGS CLI.

8. Click **Next: Confirm Order**.
9. Read and accept **Terms of Service**, and click **Create Cluster**.

Note It requires about 10 minutes to create an ACK cluster that contains multiple nodes.

Result

- After the cluster is created, you can find the cluster on the **Clusters** page in the console.
- Click **View Logs** in the **Actions** column. On the **Log Information** page, you can view log data of the cluster. To view detailed log data, click **Stack events**.
- Click **Details** in the **Actions** column. On the details page of the cluster, click the **Basic Information** tab to view basic information about the cluster. You can also click the **Connection Information** tab to view information about how to connect to the cluster.

4.3. Topology-aware CPU scheduling

Container Service for Kubernetes (ACK) provides the topology-aware CPU scheduling feature based on the new Kubernetes scheduling framework. This feature can improve the performance of CPU-sensitive workloads.

Topology-aware CPU scheduling

Multiple pods may run on a node in a Kubernetes cluster and some pods may belong to CPU-intensive workloads. In this case, pods compete for CPU resources. When this situation becomes intensive, the CPU cores that are allocated to each pod may be frequently changed. The situation intensifies when the Non-Uniform Memory Access (NUMA) nodes are used. These changes degrade the performance of the workloads. The Kubernetes CPU manager provides a CPU scheduling solution to fix this issue within a node. However, the Kubernetes CPU manager cannot find an optimal solution for allocating CPU cores within a cluster. The Kubernetes CPU manager works only on guaranteed pods and does not apply to all types of pods. In a guaranteed pod, each container is configured with requests and limits on CPU resources. In addition, the requests and limits are set to the same value.

We recommend that you use topology-aware CPU scheduling if your workloads are compute-intensive, CPU-sensitive, and run on ECS Bare Metal instances that have multiple CPU cores.

For more information, see Topology-aware CPU scheduling.

4.4. GPU scheduling

4.4.1. cGPU Professional Edition

4.4.1.1. Overview of cGPU Professional Edition

This topic introduces cGPU and describes the benefits of cGPU Professional Edition by comparing it with cGPU Basic Edition.

Benefits of cGPU Professional Edition

Benefit	Description
Supports graphics processing unit (GPU) sharing, scheduling, and memory isolation.	• Supports GPU sharing, scheduling, and memory isolation on a one-pod-one-GPU basis. This is commonly used in model inference scenarios. • Supports GPU sharing, scheduling, and memory isolation on a one-pod-multi-GPU basis. This is commonly used to build the code to train distributed models.

Benefit	Description

Supports flexible GPU sharing and memory isolation policies.	• Supports GPU allocation by using the binpack and spread algorithms.
	  ◦ Binpack: The system preferably shares one GPU with multiple pods. This applies to scenarios where high GPU utilization is required.
	  ◦ Spread: The system attempts to allocate one GPU to each pod. This applies to scenarios where the high availability of GPUs is required. The system attempts to avoid allocating the same GPU to different pod replicas of an application.
	• Supports GPU sharing without memory isolation. This applies to deep learning scenarios where applications are configured with user-defined isolation systems at the application layer.
	• Supports GPU sharing on multiple GPUs and memory isolation.

Supports comprehensive monitoring of GPU resources.	Supports monitoring of both exclusive GPUs and shared GPUs.

Comparison between cGPU Basic Edition and cGPU Professional Edition

Feature	cGPU Professional Edition	cGPU Basic Edition
GPU sharing and scheduling on one GPU	Supported	Supported
GPU sharing and scheduling on multiple GPUs	Supported	Not supported
Memory isolation on one GPU	Supported	Supported
Memory isolation on multiple GPUs	Supported	Not supported
Monitoring and auto scaling of exclusive GPUs and shared GPUs	Supported	Supported
Node pools that support flexible policy configurations	Supported. Allows you to create different GPU policies for a node pool. You can enable GPU sharing with or without memory isolation for a node pool.	Supported. You can configure different GPU policies for a node pool. You can enable GPU sharing with or without memory isolation for a node pool. In addition, you can use the binpack or spread algorithm to allocate GPUs.
Allocate GPU memory to pods by using algorithms	Supported. GPUs can be allocated by using the binpack and spread algorithms. You can choose binpack or spread to meet your business requirements.	Supported. By default, GPUs are allocated by using the binpack algorithm.

> **Note** Different cGPU editions are intended for different types of Kubernetes clusters:
>
> - cGPU Basic Edition is used after you install ack-ai-installer in a dedicated Kubernetes cluster with GPU-accelerated nodes. For more information, see Install the cGPU component.
> - cGPU Professional Edition is used after you install ack-ai-installer in a professional Kubernetes cluster with GPU-accelerated nodes. For more information, see Install and use ack-ai-installer and the GPU scheduling inspection tool.

GPU sharing solution by Alibaba Cloud

A key requirement of GPU sharing among multiple pods is to isolate the GPU memory and computing power that are allocated to each pod. When you run multiple containers on one GPU, the GPU resources are allocated to each container as required. However, if one container occupies excessive GPU resources, the performance of the other containers may be affected. To address this issue, many solutions have been developed in the computing industry. Technologies, such as NVIDIA virtual GPU (vGPU), NVIDIA Multi-Process Service (MPS), rCUDA, and vCUDA, all contribute to fine-grained GPU resource allocation.

The cGPU solution uses the server kernel driver that is developed by Alibaba Cloud to provide more efficient use of the underlying drivers of NVIDIA GPUs. cGPU provides the following features:

- High compatibility: cGPU is compatible with standard open source solutions, such as Kubernetes and NVIDIA Docker.
- Ease of use: cGPU provides excellent user experience. To replace a Compute Unified Device Architecture (CUDA) library of an AI application, you do not need to recompile the application or create a new container image.
- Stability: cGPU provides stable underlying operations on NVIDIA GPUs. API operations on CUDA libraries and some private API operations on CUDA Deep Neural Network (cuDNN) are difficult to call.
- Resource isolation: cGPU ensures that the allocated GPU memory and computing capacity do not affect each other.

cGPU provides a cost-effective, reliable, and user-friendly solution that allows you to enable GPU scheduling and memory isolation.

4.4.1.2. Install and use ack-ai-installer and the GPU scheduling inspection tool

Container Service for Kubernetes (ACK) provides graphics processing unit (GPU) sharing based on cGPU. You can use cGPU to share one GPU in model inference scenarios. In addition, the NVIDIA kernel driver ensures that the GPU memory allocated to each container is isolated from the other containers. This topic describes how to install ack-ai-installer and the GPU scheduling inspection tool on a GPU-accelerated node. You can use them to implement GPU sharing and memory isolation.

Prerequisites

- A professional managed Kubernetes cluster is created. When you create the cluster, you must select the instance types for **heterogeneous computing**, including GPU-accelerated, FPGA-accelerated, and NPU-accelerated instances. For more information about other cluster parameters, see Create a professional managed Kubernetes cluster.

  > **Note** You can install ack-ai-installer in only professional managed Kubernetes clusters. To install ack-ai-installer in dedicated Kubernetes clusters, Submit a ticket to add your account to the whitelist.

- Connect to Kubernetes clusters by using kubectl.

- You can install ack-ai-installer on nodes that are deployed in all regions. However, only regions in the following table support GPU memory isolation. If you require GPU memory isolation, make sure that the region where your cluster is deployed is included in the following table.

Region	Region ID
China (Beijing)	cn-beijing
China (Shanghai)	cn-shanghai
China (Hangzhou)	cn-hangzhou
China (Zhangjiakou)	cn-zhangjiakou
China (Shenzhen)	cn-shenzhen
China (Chengdu)	cn-chengdu
China (Heyuan)	cn-heyuan
China (Hong Kong)	cn-hongkong
Indonesia (Jakarta)	ap-southeast-5
Singapore (Singapore)	ap-southeast-1
US (Virginia)	us-east-1
US (Silicon Valley)	us-west-1

Limits

Item	Supported versions
Kubernetes	V1.18.8 and later
Helm	V3.0 and later
NVIDIA driver	V418.87.01 and later
Docker	19.03.5
Operating system	CentOS 7.6, CentOS 7.7, Ubuntu 16.04 and 18.04, and Aliyun Cloud Linux 2.x.
GPU	Tesla P4, Tesla P100, Tesla T4, and Tesla V100

Step 1: Install ack-ai-installer

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. In the upper-right corner of the **App Catalog** page, enter ack-ai-installer into the search bar and click the search icon. Find and click **ack-ai-installer**.
4. On the **App Catalog - ack-ai-installer** page, select a cluster in the Deploy section to deploy ack-ai-installer and click **Create**.

After ack-ai-installer is installed, you are redirected to the details page of ack-ai-installer. You can view the plug-ins of ack-ai-installer.

Step 2: Enable GPU sharing and memory isolation

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster where ack-ai-installer is installed and click **Node Pools** in the **Actions** column.
4. In the upper-right corner of the **Node Pools** page, click **Create Node Pool**.
5. In the **Create Node Pool** dialog box, configure the node pool.

   For more information, see Create a professional managed Kubernetes cluster. The following list describes some of the parameters:

   - Quantity: Specify the initial number of nodes in the node pool. If you do not need to create nodes in the node pool, set this parameter to 0.
   - Operating System: Select an operating system for the nodes. Supported operating systems are CentOS, Aliyun Cloud Linux 2.x, and Windows.
   - ECS Label: You can add labels to the ECS instances.
   - Custom Resource Group: You can specify the resource group to which the nodes in the node pool belong.
   - Node Label: You can add labels to the nodes. For more information about node labels, see Labels used by ACK to control GPUs.
     - Enable both GPU sharing and memory isolation.

       Click ⊕ on the right side of **Node Label**. Set **Key** to ack.node.gpu.schedule and set **Value** to cgpu.

       > **Note** If you want to enable only GPU sharing for the node pool, set **Key** to ack.node.gpu.schedule and set **Value** to share for **Node Label**.

     - Use the binpack algorithm to allocate GPUs to pods.

       Click ⊕ on the right side of **Node Label**. Set **Key** to ack.node.gpu.placement and set **Value** to binpack.
6. Click **Confirm Order**.

Step 3: Add GPU-accelerated nodes

After the node pool is created, you can add GPU-accelerated nodes to the node pool. When you add GPU-accelerated nodes, you must select the instance types for **heterogeneous computing**, including GPU-accelerated, FPGA-accelerated, and NPU-accelerated instances. For more information, see Add existing ECS instances to an ACK cluster or 管理节点池.

> **Note** If you have already added GPU-accelerated nodes to the node pool when you create the node pool, skip this step.

Step 4 (optional): Install and use the GPU scheduling inspection tool

1. Configure the kubeconfig file. For more information, see Connect to Kubernetes clusters by using kubectl.
2. Download kubectl-inspect-cgpu.
   - If you use Linux, run the following command to download kubectl-inspect-cgpu:

```
wget http://aliacs-k8s-cn-beijing.oss-cn-beijing.aliyuncs.com/gpushare/kubectl-inspect-cgpu-linux -O /usr/local/bin/kubectl-inspect-cgpu
```

- If you use macOS, run the following command to download kubectl-inspect-cgpu:

```
wget http://aliacs-k8s-cn-beijing.oss-cn-beijing.aliyuncs.com/gpushare/kubectl-inspect-cgpu-darwin -O /usr/local/bin/kubectl-inspect-cgpu
```

3. Run the following command to make kubectl-inspect-cgpu executable:

```
chmod +x /usr/local/bin/kubectl-inspect-cgpu
```

4. Run the following command to query GPU usage in the cluster:

```
kubectl inspect cgpu
```

Expected output:

```
NAME                  IPADDRESS     GPU0(Allocated/Total)  GPU Memory(GiB)
cn-shanghai.192.168.6.104  192.168.6.104  0/15                   0/15
------------------------------------------------------------------------------
Allocated/Total GPU Memory In Cluster:
0/15 (0%)
```

4.4.1.3. Enable GPU sharing

This topic describes how to deploy a YAML file to create containers that share one graphics processing unit (GPU). After you deploy the file, you can use cGPU to isolate the GPU memory that is allocated to each container. This improves GPU resource utilization.

Prerequisites

Install and use ack-ai-installer and the GPU scheduling inspection tool

Procedure

1. Run the following command to query information about GPU sharing in your cluster:

```
kubectl inspect cgpu
```

```
NAME                  IPADDRESS     GPU0(Allocated/Total)  GPU1(Allocated/Total)  GPU Memory(GiB)
cn-shanghai.192.168.0.4  192.168.0.4  0/7                    0/7                    0/14
------------------------------------------------------------------------------
Allocated/Total GPU Memory In Cluster:
0/14 (0%)
```

> **Note** To query detailed information about GPU sharing, run the **kubectl inspect cgpu -d** command.

2. Deploy the following YAML file to create containers that share one GPU:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: binpack
 labels:
  app: binpack
spec:
 replicas: 1
 serviceName: "binpack-1"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: binpack-1
 template: # define the pods specifications
  metadata:
   labels:
    app: binpack-1
  spec:
   containers:
   - name: binpack-1
    image: registry.cn-shanghai.aliyuncs.com/tensorflow-samples/tensorflow-gpu-mem:10.0-runtime-cento
s7
     command:
      - python3
      - /app/main.py
     resources:
      limits:
       # GiB
       aliyun.com/gpu-mem: 3
```

Note aliyun.com/gpu-mem: Specify the amount of memory that is allocated to the container.

3. Run the following command to query the memory usage of the GPU:

```
kubectl inspect cgpu
```

Expected output:

```
NAME                  IPADDRESS     GPU0(Allocated/Total)  GPU Memory(GiB)
cn-beijing.192.168.1.105  192.168.1.105  3/14                   3/14
--------------------------------------------------------------------------
Allocated/Total GPU Memory In Cluster:
3/14 (21%)
```

The output shows that the total GPU memory of the cn-beijing.192.168.1.105 node is 14 GB and 3 GB of GPU memory is allocated.

Result

You can use the following method to check whether cGPU has isolated the GPU memory that is allocated to different containers:

- Run the following command to view the log of the application that is deployed in Step 2.
  You can check whether GPU memory is isolated by cGPU based on the log data.

```
kubectl logs binpack-0 --tail=1
```

Expected output:

```
2020-03-13 09:14:13.931003: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1326] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 2832 MB memory) -> physical GPU (device: 0, name: Tesla T4, pci bus id: 0000:00:07.0, compute capability: 7.5)
```

The output indicates that the container requests 2,832 MiB of GPU memory.

- Run the following command to log on to the container and view the amount of GPU memory that is allocated to the container:

```
kubectl exec -it binpack-0 nvidia-smi
```

Expected output:

```
Fri Mar 13 09:32:18 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:00:07.0 Off |                    0 |
| N/A  41C    P0    26W /  70W |   3043MiB /  3231MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

The output indicates that the amount of GPU memory allocated to the container is 3,231 MiB.

- Run the following command to query the total GPU memory of the node where the application is deployed. Perform this operation on the node.

```
nvidia-smi
```

Expected output:

```
Fri Mar 13 17:36:24 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:00:07.0 Off |                    0 |
| N/A  40C    P0    26W /  70W |   3053MiB / 15079MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      8796      C   python3                                 3043MiB |
+-----------------------------------------------------------------------------+
```

The output indicates that the total GPU memory of the host is 15,079 MiB and the amount of GPU memory that is allocated to the container is 3,053 MiB.

4.4.1.4. Use cGPU to achieve GPU sharing based on multiple GPUs

You can use cGPU in professional Kubernetes clusters to achieve graphics processing unit (GPU) sharing and memory isolation. This topic describes how to use cGPU to achieve GPU sharing based on multiple GPUs.

Prerequisites

Install and use ack-ai-installer and the GPU scheduling inspection tool

Context

GPU sharing based on multiple GPUs works in the following way: an application requests N GiB of GPU memory in total and requires M GPUs to allocate the requested amount of memory. The memory that is allocated by each GPU is N/M. The value of N/M must be an integer and the used GPUs must be installed on the same node. For example, an application requests 8 GiB of GPU memory and requires that the memory is allocated by 4 GPUs. In this case, the application is allocated with 8 GiB of GPU memory from 4 GPUs that are installed on the same node. Each GPU allocates 2 GiB of memory.

Procedure

1. Create a file named *binpack-1.yaml*.

   Create a StatefulSet application named binpack-1 by using a YAML file and add the following settings to the YAML file:

   - Set the number of pod replicas to 1.
   - Declare 4 GPUs for each pod and specify that each GPU allocates 2 GiB of memory.
     - Add label `aliyun.com/gpu-count=4` to the pod.
     - Set the number of extended resources `aliyun.com/gpu-mem` to 8 for the pod.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: binpack-1
 labels:
  app: binpack-1
spec:
 # Set the number of pod replicas to 1. This makes it easier for you to check GPU memory allocation.
 replicas: 1
 serviceName: "binpack-1"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: binpack-1
 template: # define the pods specifications
  metadata:
   labels:
    app: binpack-1
    # Declare 4 GPUs for the pod and specify that each GPU allocates 2 GiB of memory.
    aliyun.com/gpu-count: "4"
  spec:
   containers:
   - name: binpack-1
    image: registry.cn-shanghai.aliyuncs.com/tensorflow-samples/tensorflow-gpu-mem:10.0-runtime-cento
s7
     resources:
      limits:
       # The pod requests 8 GiB of GPU memory in total.
       aliyun.com/gpu-mem: 8
```

2. Deploy application binpack-1.

```
kubectl apply -f binpack-1.yaml
```

Verify the result.

1. Run the following command to query the state of the pod:

```
kubectl get po
```

Expected output:

```
NAME        READY  STATUS   RESTARTS  AGE
binpack-1-0    1/1   Running  0      9m34s
```

2. Run the following command to query the number of used GPUs and the amount of memory that is allocated by each GPU:

```
kubectl exec binpack-1-0 -- nvidia-smi
```

Expected output:

```
Tue Nov  3 09:05:05 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01   Driver Version: 418.87.01   CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name      Persistence-M| Bus-Id      Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|  0  Tesla V100-SXM2...  On   | 00000000:00:07.0 Off |                    0 |
| N/A   31C    P0    54W / 300W |   2084MiB /  2150MiB |     1%      Default |
+-------------------------------+----------------------+----------------------+
|  1  Tesla V100-SXM2...  On   | 00000000:00:08.0 Off |                    0 |
| N/A   32C    P0    55W / 300W |   2084MiB /  2150MiB |     0%      Default |
+-------------------------------+----------------------+----------------------+
|  2  Tesla V100-SXM2...  On   | 00000000:00:09.0 Off |                    0 |
| N/A   32C    P0    54W / 300W |   2084MiB /  2150MiB |     0%      Default |
+-------------------------------+----------------------+----------------------+
|  3  Tesla V100-SXM2...  On   | 00000000:00:0A.0 Off |                    0 |
| N/A   34C    P0    54W / 300W |   2084MiB /  2150MiB |     0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

The preceding output shows that 4 GPUs are used and the amount of memory that is allocated by each GPU almost equals the applied amount: 2 GiB.

> **Note** The amount of memory that is allocated by each GPU may be tens to hundreds of megabytes less than the applied amount.

3. (Optional)Run the following command to query the GPUs that are used by the container:

```
kubectl exec binpack-1-0 -- env | grep NVIDIA_VISIBLE_DEVICES
```

Expected output:

```
NVIDIA_VISIBLE_DEVICES=GPU-c09d11f5-a565-1f83-903c-a65a7ba138af,GPU-959873c5-b57b-5d24-f3aa-e700c5d431cb,GPU-abdcd4da-7505-c267-fb4d-30478a360e07,GPU-ba997778-f8ce-e433-3c84-6d15392a06a1
```

4. (Optional)You can use environment variable `ALIYUN_COM_GPU_MEM_CONTAINER` to query the amount of memory that can be allocated from each GPU to the container.

```
kubectl exec binpack-1-0 -- env | grep ALIYUN
```

Expected output:

```
ALIYUN_COM_GPU_MEM_CONTAINER=2 # The amount of memory that can be allocated from each GPU to the container is 2 GiB. The application that is deployed in the container can read the value to set memory limit for the container.
ALIYUN_COM_GPU_MEM_DEV=15 # The total memory of each GPU is 15 GiB.
```

4.4.1.5. Use node pools to control cGPU

You can use node pools to regulate the graphics processing unit (GPU) sharing and memory isolation policies of cGPU in professional managed Kubernetes clusters. This topic describes how to use node pools to control cGPU.

Prerequisites

- Helm 3.0.0 or later is used.
- A professional managed Kubernetes cluster is created.
  When you create the cluster, set the Kubernetes version and instance types based on the following description. For more information about other cluster parameters, see Create a professional managed Kubernetes cluster.

  > **Note** You can install cGPU in only professional managed Kubernetes clusters. If you want to install cGPU in dedicated Kubernetes clusters, Submit a ticket to add your account to the whitelist.

  - The Kubernetes version must be V1.18.8 or later.
  - You must select the instance types for **heterogeneous computing**, including GPU-accelerated, FPGA-accelerated, and NPU-accelerated instances.
- Only regions in the following table support GPU memory isolation. If you require GPU memory isolation, make sure that the region where your cluster is deployed is included in the following table.

Region	Region ID
China (Beijing)	cn-beijing
China (Shanghai)	cn-shanghai
China (Hangzhou)	cn-hangzhou
China (Zhangjiakou)	cn-zhangjiakou
China (Shenzhen)	cn-shenzhen
China (Chengdu)	cn-chengdu
China (Heyuan)	cn-heyuan
China (Hong Kong)	cn-hongkong
Indonesia (Jakarta)	ap-southeast-5
Singapore (Singapore)	ap-southeast-1
US (Virginia)	us-east-1
US (Silicon Valley)	us-west-1

- Node pools are created with names and labels that are set as required to implement GPU sharing and memory isolation.
  You can customize the names of the node pools. In this example, the node pools are named cgpu and cgpu-no-isolation. For more information, see Labels used by ACK to control GPUs.

Node pool name	GPU sharing	Memory isolation	Node Label

Node pool name	GPU sharing	Memory isolation	Node Label
cgpu-no-isolation	Supported	Not supported	◦ ack.node.gpu.schedule=share ◦ ack.node.gpu.placement=binpack
cgpu	Supported	Supported	◦ ack.node.gpu.schedule=cgpu ◦ ack.node.gpu.placement=binpack

Scenarios

When you use cGPU in a cluster of Container Service for Kubernetes (ACK), the following scenarios may exist at the same time:

- The amount of GPU memory that can be allocated to Job A is already specified in the script. In this case, the ACK cluster only needs to enable GPU sharing for Job A. No memory isolation is required.
- The amount of GPU memory that can be allocated to Job B is not specified in the script. In this case, the ACK cluster must enable both GPU sharing and memory isolation for Job B.

To support both scenarios as the same time in an ACK cluster, you can use node pools to control cGPU. You only need to create two node pools in the cluster:

- Create a node pool that supports only GPU sharing. Do not enable memory isolation. This node pool is used to run Job A.
- Create another node pool that supports both GPU sharing and memory isolation. This node pool is used to run Job B.

Considerations

When you use node pools to control cGPU, take note of the following limits:

- We recommend that you configure a node selector for each job.
  When you control cGPU based on node pools, if a job is not configured with a node selector, the pods of the job may be scheduled to other node pools. For example, job pods that do not require memory isolation may be scheduled to nodes that support memory isolation.
- To disable memory isolation for a node, uninstall the kernel module of cGPU on the node and restart the instance.
  - If label ack.node.gpu.schedule=cgpu is changed to ack.node.gpu.schedule=share on a node, memory isolation is not disabled for the node. You must uninstall the kernel module of cGPU on the node and restart the instance. For more information about how to uninstall the kernel module of cGPU on a node, see Install and use the cGPU service by using Docker.
  - If label ack.node.gpu.schedule=share is changed to ack.node.gpu.schedule=cgpu on a node, memory isolation is enabled for the node.

Step 1: Install ack-ai-installer

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. In the upper-right corner of the **App Catalog** page, enter ack-ai-installer into the search bar and click the search icon. Find and click **ack-ai-installer**.
4. On the **App Catalog - ack-ai-installer** page, select a cluster in the Deploy section to deploy ack-ai-

installer and click **Create**.
After ack-ai-installer is installed, you are redirected to the details page of ack-ai-installer. You can view the plug-ins of ack-ai-installer.

Step 2: Create node pools

Create a node pool that supports both GPU sharing and memory isolation. Create another node pool that supports only GPU sharing.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster where cGPU is deployed and click **Node Pools** in the **Actions** column.
4. In the upper-right corner of the **Node Pools** page, click **Create Node Pool**.
5. In the **Create Node Pool** dialog box, configure the node pool.

   For more information, see Create a professional managed Kubernetes cluster. The following list describes some of the parameters:

   - Name: Set **Name** to cgpu.
   - Quantity: Specify the initial number of nodes in the node pool. If you do not need to create nodes in the node pool, set this parameter to 0.
   - Operating System: Select an operating system for the nodes. Supported operating systems are CentOS 7.x, Alibaba Cloud Linux 2.x.
   - ECS Label: You can add labels to the ECS instances.
   - Custom Resource Group: You can specify the resource group to which the nodes in the node pool belong.
   - Node Label: You can add labels to the nodes.
     - Enable both GPU sharing and memory isolation.
       Click ⊕ on the right side of **Node Label**. Set **Key** to ack.node.gpu.schedule and set **Value** to cgpu.
     - Use the binpack algorithm to allocate GPUs to pods.
       Click ⊕ on the right side of **Node Label**. Set **Key** to ack.node.gpu.placement and set **Value** to binpack.
6. Repeat Step and Step to create node pool cgpu-no-isolation.

   When you create node pool cgpu-no-isolation, set the name to cgpu-no-isolation and add labels ack.node.gpu.schedule=share and ack.node.gpu.placement=binpack to the nodes in the node pool.

Step 3: Add GPU-accelerated nodes

After the node pool is created, you can add GPU-accelerated nodes to the node pool. When you add GPU-accelerated nodes, you must select the instance types for **heterogeneous computing**, including GPU-accelerated, FPGA-accelerated, and NPU-accelerated instances. For more information, see Add existing ECS instances to an ACK cluster or 管理节点池.

> **Note** If you have already added GPU-accelerated nodes to the node pool when you create the node pool, skip this step.

Step 4: Submit jobs

To check whether GPU sharing and memory isolation are enabled as required for the node pools, submit two jobs:

- cgpu-test: The amount of GPU memory to be allocated to this job is not specified in the script of the job. Therefore, memory isolation is required to run this job without errors.
- cgpu-test-no-isolation: The amount of memory to be allocated to this job per GPU is specified in the script of the job. Therefore, memory isolation is not required.

Submit job cgpu-test-no-isolation

1. Create a file named *cgpu-test-no-isolation.yaml*.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: cgpu-test-no-isolation
 labels:
  app: cgpu-test-no-isolation
spec:
 replicas: 1
 serviceName: "cgpu-test-no-isolation"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: cgpu-test-no-isolation
 template: # define the pods specifications
  metadata:
   labels:
    app: cgpu-test-no-isolation
  spec:
   nodeSelector:
    # Add a node selector to schedule the job to matched nodes.
    # Node pool cgpu-no-isolation consists of nodes that are labeled with ack.node.gpu.schedule=share.
    ack.node.gpu.schedule: "share"
   containers:
   - name: cgpu-test-no-isolation
    image: cheyang/gpu-player:v2
    resources:
     limits:
      # Request 3 GiB of GPU memory.
      aliyun.com/gpu-mem: 3
```

   - aliyun.com/gpu-mem: Specify the amount of GPU memory.
   - nodeSelector: Select node pool cgpu-no-isolation.
2. Submit job cgpu-test-no-isolation

```
kubectl apply -f cgpu-test-no-isolation.yaml
```

Submit job cgpu-test

1. Create a file named *cgpu-test.yaml*.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: cgpu-test
 labels:
  app: cgpu-test
spec:
 replicas: 1
 serviceName: "cgpu-test"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: cgpu-test
 template: # define the pods specifications
  metadata:
   labels:
    app: cgpu-test
  spec:
   nodeSelector:
    # Add a node selector and select node pool cgpu.
    ack.node.gpu.schedule: "cgpu"
   containers:
   - name: cgpu-test
    image: registry.cn-shanghai.aliyuncs.com/tensorflow-samples/tensorflow-gpu-mem:10.0-runtime-cento
s7
     command:
      - python3
      - /app/main.py
     env:
     resources:
      limits:
       # Request 3 GiB of GPU memory.
       aliyun.com/gpu-mem: 3
```

- aliyun.com/gpu-mem: Specify the amount of GPU memory.
- nodeSelector: Select node pool cgpu.

2. Submit job cgpu-test

```
kubectl apply -f cgpu-test.yaml
```

Step 5: Verify the job execution results

If memory isolation is enabled, the amount of GPU memory that a container can use equals the amount that is allocated from the GPU. If memory isolation is disabled, the amount of GPU memory that a container can use equals the total memory of the GPU. Run the following command to query the amount of GPU memory that can be used by the job containers:

- Run the following command to query the job pods:

```
kubectl get po
```

Expected output:

```
NAME           READY  STATUS   RESTARTS  AGE
cgpu-test-0        1/1   Running  0     5m55s
cgpu-test-no-isolation-0  1/1   Running  0     6m42s
```

- Run the following command to query the amount of GPU memory that can be used by container cgpu-test-0:

```
kubectl exec cgpu-test-0 nvidia-smi
```

Expected output:

```
Mon Nov  2 11:33:10 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:07.0 Off |                    0 |
| N/A   34C    P0    54W / 300W |   3039MiB /  3226MiB |     1%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

The preceding output shows that 3,226 MiB of GPU memory can be used by container cgpu-test-0. The total GPU memory is 16 GiB. This means that GPU memory isolation is enabled.

- Run the following command to query the amount of GPU memory that can be used by container cgpu-test-no-isolation-0:

```
kubectl exec cgpu-test-no-isolation-0 nvidia-smi
```

Expected output:

```
Mon Nov  2 11:39:59 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:07.0 Off |                    0 |
| N/A   37C    P0    56W / 300W |   1929MiB / 16130MiB |     1%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

The preceding output shows that 16,130 MiB of GPU memory can be used by container cgpu-test-no-isolation-0. The total GPU memory is 16 GiB. This means that GPU memory isolation is disabled.
Compare the results returned from container cgpu-test-no-isolation-0 and container cgpu-test-0, you can find that container cgpu-test-no-isolation-0 is allocated with the total amount of GPU memory and container cgpu-test-0 is allocated with only the applied amount of GPU memory. This means that you succeed to use node pools to control cGPU for GPU sharing and memory isolation.

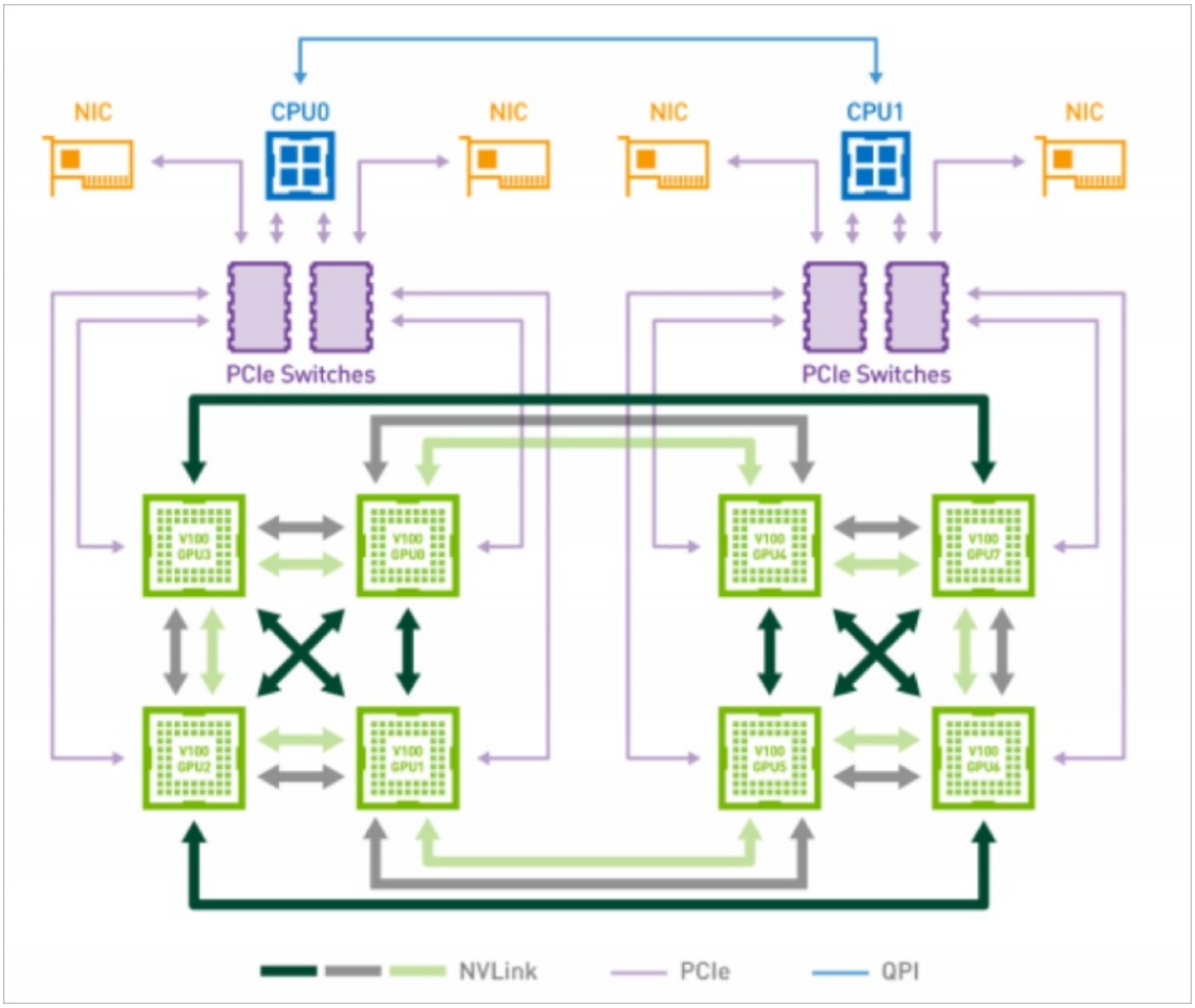
4.4.2. GPU topology-aware scheduling

4.4.2.1. Overview

This topic describes the GPU topology. It also includes further details of the benefits of GPU topology-aware scheduling.

GPU topology

The following figure shows the topology of eight Tesla V100 GPUs that communicate with each other through NVLinks. Each Tesla V100 GPU is assigned six NVLinks. However, NVLinks cannot be established between every two Tesla V100 GPUs. At most two NVLinks can be established between two Tesla V100 GPUs. In this example, two NVLinks are established between GPU 0 and GPU 3. Two NVLinks are established between GPU 0 and GPU 4. One NVLink is established between GPU 0 and GPU 1. GPU 0 and GPU 6 communicate with each other through Peripheral Component Interconnect Express (PCIe), instead of NVLinks.



Benefits

The one-way communication bandwidth of an NVLink is 25 Gbit/s. The two-way communication bandwidth of an NVLink is 50 Gbit/s. The PCIe bandwidth is 16 Gbit/s. In a training job, the training speed depends on the different combinations of GPUs. Therefore, the optimal combination of GPUs can be selected during the GPU scheduling process. This ensures the optimal training speed.

Kubernetes does not support GPU topology-aware scheduling. In this case, GPUs are selected at random. The training speed can vary based on different combinations of GPUs. To fix this issue, Container Service for Kubernetes (ACK) supports GPU topology-aware scheduling based on the scheduling framework. You can use this feature to select a combination of GPUs on GPU nodes. This ensures the optimal training speed.

Related information

- Install the ack-ai-installer component
- Use topology-aware GPU scheduling to achieve optimal GPU acceleration for TensorFlow distributed jobs

4.4.2.2. Install the ack-ai-installer component

This topic describes the components and configurations that are required to activate GPU topology-aware scheduling.

Prerequisites

- Create a professional managed Kubernetes cluster. Set Instance Type of the cluster to **Heterogeneous Computing**. For more information, see Create a professional managed Kubernetes cluster.

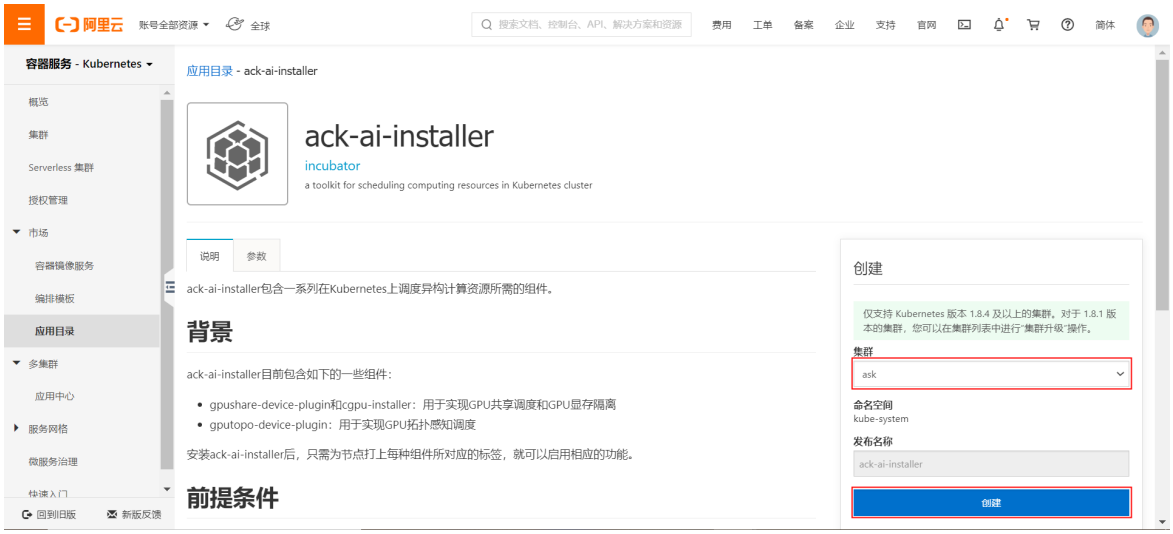
  > **Notice** Only professional managed Kubernetes cluster are supported. If you want to activate GPU topology-aware scheduling for dedicated Kubernetes clusters, submit a ticket to add your account to the whitelist.

- Use kubectl to connect to the Container Service for Kubernetes (ACK) cluster. For more information, see Connect to Kubernetes clusters by using kubectl.
- The following table lists the required components and versions.

Component	Required version
Kubernetes	V1.18.8 and later
Helm	V3.0 and later
Nvidia	V418.87.01 and later
NVIDIA Collective Communications Library (NCCL)	2.7+
Docker	19.03.5
Operating system	CentOS 7.6, CentOS 7.7, Ubuntu 16.04, Ubuntu 18.04, and Alibaba Cloud Linux 2
Graphics card	V100

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, select **Name** from the drop-down list in the upper-right corner of the page, enter *ack-ai-installer* in the search box, and then click the search icon.
4. In the **Deploy** pane of the **ack-ai-installer** page, select the cluster that you want to manage from the **Cluster** drop-down list and click **Create**.



Related information

- Overview
- Use topology-aware GPU scheduling to achieve optimal GPU acceleration for TensorFlow distributed jobs

4.4.2.3. Use topology-aware GPU scheduling to achieve optimal GPU acceleration for TensorFlow distributed jobs

Container Service for Kubernetes (ACK) supports topology-aware GPU scheduling based on the scheduling framework. This feature selects a combination of GPUs from GPU-accelerated nodes to achieve optimal GPU acceleration for training jobs. This topic describes how to use topology-aware GPU scheduling to achieve optimal GPU acceleration for TensorFlow distributed jobs.

Prerequisites

- Create a professional managed Kubernetes cluster.
- Arena is installed.
- Install the ack-ai-installer component.
- The following table lists the required components and versions.

Component	Version
Kubernetes	V1.18.8 and later
Helm	V3.0 and later
Nvidia	V418.87.01 and later
NVIDIA Collective Communications Library (NCCL)	2.7+
Docker	19.03.5
Operating system	CentOS 7.6, CentOS 7.7, Ubuntu 16.04 and 18.04, and Alibaba Cloud Linux 2.
GPU	V100

Limits

- Topology-aware GPU scheduling is applicable to only Message Passing Interface (MPI) jobs that are trained by using a distributed framework.
- The resources that are requested by pods must meet specific requirements before the pods can be created to submit and start jobs. Otherwise, the requests remain pending for resources.

Procedure

Configure nodes

Run the following command to set the node label and explicitly enable topology-aware GPU scheduling for nodes:

```
kubectl label node <Your Node Name> ack.node.gpu.schedule=topology
```

Note After topology-aware GPU scheduling is enabled on nodes, common GPU scheduling is no longer supported. You can run the following command to change the label and resume common GPU scheduling.

```
kubectl label node <Your Node Name> ack.node.gpu.schedule=default --overwrite
```

Submit a job

Submit an MPI job and set --gputopology to `true`.

```
arena submit mpi --gputopology=true ***
```

Example 1: Train VGG16

Note In this topic, two servers are deployed in the test cluster. Each server has eight V100 GPUs.

Use topology-aware GPU scheduling to train VGG16

1. Run the following command to submit a job to the cluster:

```
arena submit mpi \
 --name=tensorflow-topo-4-vgg16 \
 --gpus=1 \
 --workers=4 \
 --gputopology=true \
 --image=registry.cn-hangzhou.aliyuncs.com/kubernetes-image-hub/tensorflow-benchmark:tf2.3.0-py3.7-cuda10.1 \
 "mpirun --allow-run-as-root -np "4" -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=eth0 -x LD_LIBRARY_PATH -x PATH --mca pml ob1 --mca btl_tcp_if_include eth0 --mca oob_tcp_if_include eth0 --mca orte_keep_fqdn_hostnames t --mca btl ^openib python /tensorflow/benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model=vgg16 --batch_size=64 --variable_update=horovod"
```

2. Run the following command to query the state of the job:

```
arena get tensorflow-topo-4-vgg16 --type mpijob
```

Expected output:

```
Name:    tensorflow-topo-4-vgg16
Status:  RUNNINGNamespace: default
Priority: N/A
Trainer:  MPIJOB
Duration: 2m
Instances:
 NAME                    STATUS  AGE IS_CHIEF GPU(Requested) NODE
 ----                    ------  --- -------- -------------- ----
 tensorflow-topo-4-vgg16-launcher-lmhjl Running 2m  true    0        cn-shanghai.192.168.16.172
 tensorflow-topo-4-vgg16-worker-0     Running 2m  false   1        cn-shanghai.192.168.16.173
 tensorflow-topo-4-vgg16-worker-1     Running 2m  false   1        cn-shanghai.192.168.16.173
 tensorflow-topo-4-vgg16-worker-2     Running 2m  false   1        cn-shanghai.192.168.16.173
 tensorflow-topo-4-vgg16-worker-3     Running 2m  false   1        cn-shanghai.192.168.16.173
```

3. Run the following command to print the job log:

```
arena logs -f tensorflow-topo-4-vgg16
```

Expected output:

```
total images/sec: 991.92
```

Use common GPU scheduling to train VGG16

1. Run the following command to submit a job to the cluster:

```
arena submit mpi \
 --name=tensorflow-4-vgg16 \
 --gpus=1 \
 --workers=4 \
 --image=registry.cn-hangzhou.aliyuncs.com/kubernetes-image-hub/tensorflow-benchmark:tf2.3.0-py3.7-cuda10.1 \
 "mpirun --allow-run-as-root -np "4" -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=eth0 -x LD_LIBRARY_PATH -x PATH --mca pml ob1 --mca btl_tcp_if_include eth0 --mca oob_tcp_if_include eth0 --mca orte_keep_fqdn_hostnames t --mca btl ^openib python /tensorflow/benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model=vgg16 --batch_size=64 --variable_update=horovod"
```

2. Run the following command to query the state of the job:

```
arena get tensorflow-4-vgg16 --type mpijob
```

Expected output:

```
Name:    tensorflow-4-vgg16
Status:  RUNNING
Namespace: default
Priority: N/A
Trainer:  MPIJOB
Duration: 9s
Instances:
 NAME                    STATUS  AGE IS_CHIEF GPU(Requested) NODE
 ----                    ------  --- -------- -------------- ----
 tensorflow-4-vgg16-launcher-xc28k Running 9s  true    0        cn-shanghai.192.168.16.172
 tensorflow-4-vgg16-worker-0     Running 9s  false   1        cn-shanghai.192.168.16.172
 tensorflow-4-vgg16-worker-1     Running 9s  false   1        cn-shanghai.192.168.16.173
 tensorflow-4-vgg16-worker-2     Running 9s  false   1        cn-shanghai.192.168.16.172
 tensorflow-4-vgg16-worker-3     Running 9s  false   1        cn-shanghai.192.168.16.173
```

3. Run the following command to print the job log:

```
arena logs -f tensorflow-4-vgg16
```

Expected output:

```
total images/sec: 200.47
```

Example 2: Train ResNet50

Use topology-aware GPU scheduling to train ResNet50

1. Run the following command to submit a job to the cluster:

```
arena submit mpi \
 --name=tensorflow-topo-4-resnet50 \
 --gpus=1 \
 --workers=4 \
 --gputopology=true \
 --image=registry.cn-hangzhou.aliyuncs.com/kubernetes-image-hub/tensorflow-benchmark:tf2.3.0-py3.7-cuda10.1 \
 "mpirun --allow-run-as-root -np "4" -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=eth0 -x LD_LIBRARY_PATH -x PATH --mca pml ob1 --mca btl_tcp_if_include eth0 --mca oob_tcp_if_include eth0 --mca orte_keep_fqdn_hostnames t --mca btl ^openib python /tensorflow/benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model=resnet50 --batch_size=64  --variable_update=horovod"
```

2. Run the following command to query the state of the job:

```
arena get tensorflow-topo-4-resnet50 --type mpijob
```

Expected output:

```
Name:    tensorflow-topo-4-resnet50
Status:  RUNNING
Namespace: default
Priority: N/A
Trainer:  MPIJOB
Duration: 8s
Instances:
 NAME                    STATUS  AGE IS_CHIEF GPU(Requested) NODE
 ----                    ------  --- -------- -------------- ----
 tensorflow-topo-4-resnet50-launcher-7ln8j Running 8s  true    0       cn-shanghai.192.168.16.172
 tensorflow-topo-4-resnet50-worker-0     Running 8s  false   1       cn-shanghai.192.168.16.173
 tensorflow-topo-4-resnet50-worker-1     Running 8s  false   1       cn-shanghai.192.168.16.173
 tensorflow-topo-4-resnet50-worker-2     Running 8s  false   1       cn-shanghai.192.168.16.173
 tensorflow-topo-4-resnet50-worker-3     Running 8s  false   1       cn-shanghai.192.168.16.173
```

3. Run the following command to print the job log:

```
arena logs -f tensorflow-topo-4-resnet50
```

Expected output:

```
total images/sec: 1471.55
```

Use common GPU scheduling to train ResNet50

1. Run the following command to submit a job to the cluster:

```
arena submit mpi \
 --name=tensorflow-4-resnet50 \
 --gpus=1 \
 --workers=4 \
 --image=registry.cn-hangzhou.aliyuncs.com/kubernetes-image-hub/tensorflow-benchmark:tf2.3.0-py3.7-cuda10.1 \
 "mpirun --allow-run-as-root -np "4" -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=eth0 -x LD_LIBRARY_PATH -x PATH --mca pml ob1 --mca btl_tcp_if_include eth0 --mca oob_tcp_if_include eth0 --mca orte_keep_fqdn_hostnames t --mca btl ^openib python /tensorflow/benchmarks/scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --model=resnet50 --batch_size=64  --variable_update=horovod"
```

2. Run the following command to query the state of the job:

```
arena get tensorflow-4-resnet50 --type mpijob
```

Expected output:

```
Name:    tensorflow-4-resnet50
Status:  RUNNING
Namespace: default
Priority: N/A
Trainer:  MPIJOB
Duration: 9s
Instances:
 NAME                  STATUS  AGE IS_CHIEF GPU(Requested) NODE
 ----                  ------  --- -------- -------------- ----
 tensorflow-4-resnet50-launcher-q24hv Running 9s  true    0        cn-shanghai.192.168.16.172
 tensorflow-4-resnet50-worker-0     Running 9s  false   1        cn-shanghai.192.168.16.172
 tensorflow-4-resnet50-worker-1     Running 9s  false   1        cn-shanghai.192.168.16.173
 tensorflow-4-resnet50-worker-2     Running 9s  false   1        cn-shanghai.192.168.16.172
 tensorflow-4-resnet50-worker-3     Running 9s  false   1        cn-shanghai.192.168.16.173
```

3. Run the following command to print the job log:

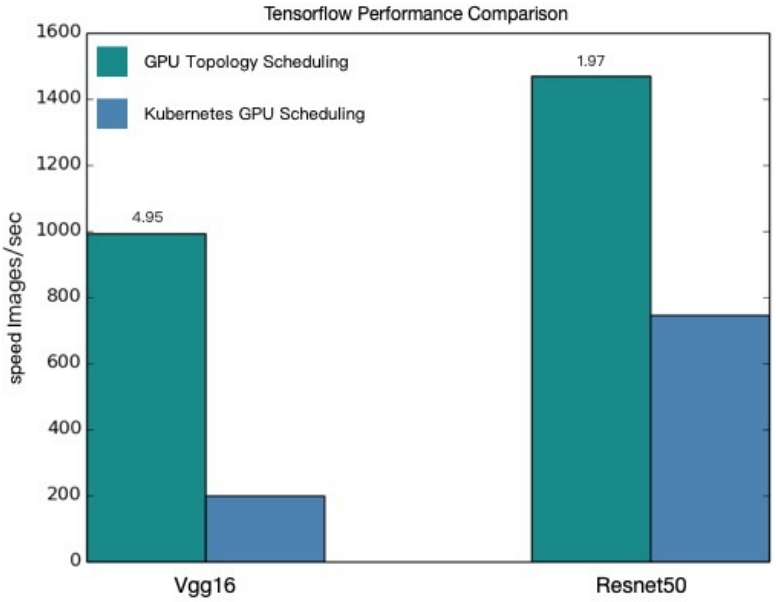
```
arena logs -f tensorflow-4-resnet50
```

Expected output:

```
total images/sec: 745.38
```

Performance comparison

The following figure shows the performance difference between topology-aware GPU scheduling and common GPU scheduling based on the preceding examples.



The figure shows that after topology-aware GPU scheduling is enabled, the TensorFlow distributed jobs are significantly accelerated.

> **Note** The improvement that is achieved by topology-aware GPU scheduling varies based on the models that you use and the cluster environment. You can evaluate the performance of your models based on the preceding examples.

Related information

- Overview
- Install the ack-ai-installer component

4.4.2.4. Use GPU topology-aware scheduling to achieve optimal GPU acceleration for PyTorch distributed jobs

Based on the scheduling framework, Container Service for Kubernetes (ACK) supports GPU topology-aware scheduling. This feature selects a combination of GPUs from GPU-accelerated nodes to achieve optimal GPU acceleration for training jobs. This topic describes how to use GPU topology-aware scheduling to achieve optimal GPU acceleration for PyTorch distributed jobs.

Prerequisites

- Create a professional managed Kubernetes cluster.
- Arena is installed.
- Install the ack-ai-installer component.
- The following table lists the required components and versions.

Component	Version
Kubernetes	V1.18.8 and later

Component	Version
Helm	V3.0 and later
Nvidia	V418.87.01 and later
NVIDIA Collective Communications Library (NCCL)	2.7+
Docker	19.03.5
Operating system	CentOS 7.6, CentOS 7.7, Ubuntu 16.04 and 18.04, and Alibaba Cloud Linux 2.
GPU	V100

Limits

- Topology-aware GPU scheduling is applicable to only Message Passing Interface (MPI) jobs that are trained by using a distributed framework.
- The resources that are requested by pods must meet specific requirements before the pods can be created to submit and start jobs. Otherwise, the requests remain pending for resources.

Procedure

Configure nodes

Run the following command to set the node label and explicitly enable GPU topology-aware scheduling for nodes:

```
kubectl label node <Your Node Name> ack.node.gpu.schedule=topology
```

Note After GPU topology-aware scheduling is enabled on nodes, common GPU scheduling is no longer supported. You can run the following command to change the label and resume common GPU scheduling.

```
kubectl label node <Your Node Name> ack.node.gpu.schedule=default --overwrite
```

Submit a job

Submit a Message Passing Interface (MPI) job and set --gputopology to `true`.

```
arena submit mpi --gputopology=true ***
```

Example 1: Train VGG16

Note In this topic, two servers are deployed in the test cluster. Each server has eight V100 GPUs.

Use GPU topology-aware scheduling to train VGG16

1. Run the following command to submit a job to the cluster:

```
arena submit mpi \
 --name=pytorch-topo-4-vgg16 \
 --gpus=1 \
 --workers=4 \
 --gputopology=true \
 --image=registry.cn-hangzhou.aliyuncs.com/kubernetes-image-hub/pytorch-benchmark:torch1.6.0-py3.7-cuda10.1 \
 "mpirun --allow-run-as-root -np "4" -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=eth0 -x LD_LIBRARY_PATH -x PATH --mca pml ob1 --mca btl_tcp_if_include eth0 --mca oob_tcp_if_include eth0 --mca orte_keep_fqdn_hostnames t --mca btl ^openib python /examples/pytorch_synthetic_benchmark.py --model=vgg16 --batch-size=64"
```

2. Run the following command to query the state of the job:

```
arena get pytorch-topo-4-vgg16 --type mpijob
```

Expected output:

```
Name:    pytorch-topo-4-vgg16
Status:  RUNNING
Namespace: default
Priority: N/A
Trainer: MPIJOB
Duration: 11s
Instances:
 NAME                  STATUS  AGE IS_CHIEF GPU(Requested) NODE
 ----                  ------  --- -------- -------------- ----
 pytorch-topo-4-vgg16-launcher-mnjzr Running 11s true    0        cn-shanghai.192.168.16.173
 pytorch-topo-4-vgg16-worker-0     Running 11s false   1        cn-shanghai.192.168.16.173
 pytorch-topo-4-vgg16-worker-1     Running 11s false   1        cn-shanghai.192.168.16.173
 pytorch-topo-4-vgg16-worker-2     Running 11s false   1        cn-shanghai.192.168.16.173
 pytorch-topo-4-vgg16-worker-3     Running 11s false   1        cn-shanghai.192.168.16.173
```

3. Run the following command to print the job log:

```
arena logs -f pytorch-topo-4-vgg16
```

Expected output:

```
Model: vgg16
Batch size: 64
Number of GPUs: 4
Running warmup...
Running benchmark...
Iter #0: 205.5 img/sec per GPU
Iter #1: 205.2 img/sec per GPU
Iter #2: 205.1 img/sec per GPU
Iter #3: 205.5 img/sec per GPU
Iter #4: 205.1 img/sec per GPU
Iter #5: 205.1 img/sec per GPU
Iter #6: 205.3 img/sec per GPU
Iter #7: 204.3 img/sec per GPU
Iter #8: 205.0 img/sec per GPU
Iter #9: 204.9 img/sec per GPU
Img/sec per GPU: 205.1 +-0.6
Total img/sec on 4 GPU(s): 820.5 +-2.5
```

Use common GPU scheduling to train VGG16

1. Run the following command to submit a job to the cluster:

```
arena submit mpi \
 --name=pytorch-4-vgg16 \
 --gpus=1 \
 --workers=4 \
 --image=registry.cn-hangzhou.aliyuncs.com/kubernetes-image-hub/pytorch-benchmark:torch1.6.0-py3.7-cuda10.1 \
 "mpirun --allow-run-as-root -np "4" -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=eth0 -x LD_LIBRARY_PATH -x PATH --mca pml ob1 --mca btl_tcp_if_include eth0 --mca oob_tcp_if_include eth0 --mca orte_keep_fqdn_hostnames t --mca btl ^openib python /examples/pytorch_synthetic_benchmark.py --model=vgg16 --batch-size=64"
```

2. Run the following command to query the state of the job:

```
arena get pytorch-4-vgg16 --type mpijob
```

Expected output:

```
Name:    pytorch-4-vgg16
Status:  RUNNING
Namespace: default
Priority: N/A
Trainer: MPIJOB
Duration: 10s
Instances:
 NAME                 STATUS  AGE IS_CHIEF GPU(Requested) NODE
 ----                 ------  --- -------- -------------- ----
 pytorch-4-vgg16-launcher-qhnxl Running 10s true    0        cn-shanghai.192.168.16.173
 pytorch-4-vgg16-worker-0     Running 10s false   1        cn-shanghai.192.168.16.173
 pytorch-4-vgg16-worker-1     Running 10s false   1        cn-shanghai.192.168.16.173
 pytorch-4-vgg16-worker-2     Running 10s false   1        cn-shanghai.192.168.16.173
 pytorch-4-vgg16-worker-3     Running 10s false   1        cn-shanghai.192.168.16.173
```

3. Run the following command to print the job log:

```
arena logs -f pytorch-4-vgg16
```

Expected output:

```
Model: vgg16
Batch size: 64
Number of GPUs: 4
Running warmup...
Running benchmark...
Iter #0: 113.1 img/sec per GPU
Iter #1: 109.5 img/sec per GPU
Iter #2: 106.5 img/sec per GPU
Iter #3: 108.5 img/sec per GPU
Iter #4: 108.1 img/sec per GPU
Iter #5: 111.2 img/sec per GPU
Iter #6: 110.7 img/sec per GPU
Iter #7: 109.8 img/sec per GPU
Iter #8: 102.8 img/sec per GPU
Iter #9: 107.9 img/sec per GPU
Img/sec per GPU: 108.8 +-5.3
Total img/sec on 4 GPU(s): 435.2 +-21.1
```

Example 2: Train ResNet50

Use GPU topology-aware scheduling to train ResNet50

1. Run the following command to submit a job to the cluster:

```
arena submit mpi \
 --name=pytorch-topo-4-resnet50 \
 --gpus=1 \
 --workers=4 \
 --gputopology=true \
 --image=registry.cn-hangzhou.aliyuncs.com/kubernetes-image-hub/pytorch-benchmark:torch1.6.0-py3.7-cuda10.1 \
 "mpirun --allow-run-as-root -np "4" -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=eth0 -x LD_LIBRARY_PATH -x PATH --mca pml ob1 --mca btl_tcp_if_include eth0 --mca oob_tcp_if_include eth0 --mca orte_keep_fqdn_hostnames t --mca btl ^openib python /examples/pytorch_synthetic_benchmark.py --model=resnet50 --batch-size=64"
```

2. Run the following command to query the state of the job:

```
arena get pytorch-topo-4-resnet50 --type mpijob
```

Expected output:

```
Name:    pytorch-topo-4-resnet50
Status:  RUNNING
Namespace: default
Priority: N/A
Trainer:  MPIJOB
Duration: 8s
Instances:
 NAME                   STATUS  AGE IS_CHIEF GPU(Requested) NODE
 ----                   ------  --- -------- -------------- ----
 pytorch-topo-4-resnet50-launcher-x7r2n Running 8s  true    0        cn-shanghai.192.168.16.173
 pytorch-topo-4-resnet50-worker-0     Running 8s  false   1        cn-shanghai.192.168.16.173
 pytorch-topo-4-resnet50-worker-1     Running 8s  false   1        cn-shanghai.192.168.16.173
 pytorch-topo-4-resnet50-worker-2     Running 8s  false   1        cn-shanghai.192.168.16.173
 pytorch-topo-4-resnet50-worker-3     Running 8s  false   1        cn-shanghai.192.168.16.173
```

3. Run the following command to print the job log:

```
arena logs -f pytorch-topo-4-resnet50
```

Expected output:

```
Model: resnet50
Batch size: 64
Number of GPUs: 4
Running warmup...
Running benchmark...
Iter #0: 331.0 img/sec per GPU
Iter #1: 330.6 img/sec per GPU
Iter #2: 330.9 img/sec per GPU
Iter #3: 330.4 img/sec per GPU
Iter #4: 330.7 img/sec per GPU
Iter #5: 330.8 img/sec per GPU
Iter #6: 329.9 img/sec per GPU
Iter #7: 330.5 img/sec per GPU
Iter #8: 330.4 img/sec per GPU
Iter #9: 329.7 img/sec per GPU
Img/sec per GPU: 330.5 +-0.8
Total img/sec on 4 GPU(s): 1321.9 +-3.2
```

Use common GPU scheduling to train ResNet50

1. Run the following command to submit a job to the cluster:

```
arena submit mpi \
 --name=pytorch-4-resnet50 \
 --gpus=1 \
 --workers=4 \
 --image=registry.cn-hangzhou.aliyuncs.com/kubernetes-image-hub/pytorch-benchmark:torch1.6.0-py3.7-cuda10.1 \
 "mpirun --allow-run-as-root -np "4" -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x NCCL_SOCKET_IFNAME=eth0 -x LD_LIBRARY_PATH -x PATH --mca pml ob1 --mca btl_tcp_if_include eth0 --mca oob_tcp_if_include eth0 --mca orte_keep_fqdn_hostnames t --mca btl ^openib python /examples/pytorch_synthetic_benchmark.py --model=resnet50 --batch-size=64"
```

2. Run the following command to query the state of the job:

```
arena get pytorch-4-resnet50 --type mpijob
```

Expected output:

```
Name:    pytorch-4-resnet50
Status:  RUNNING
Namespace: default
Priority: N/A
Trainer: MPIJOB
Duration: 10s
Instances:
 NAME                 STATUS  AGE IS_CHIEF GPU(Requested) NODE
 ----                 ------  --- -------- -------------- ----
 pytorch-4-resnet50-launcher-qw5k6 Running 10s true    0        cn-shanghai.192.168.16.173
 pytorch-4-resnet50-worker-0     Running 10s false   1        cn-shanghai.192.168.16.173
 pytorch-4-resnet50-worker-1     Running 10s false   1        cn-shanghai.192.168.16.173
 pytorch-4-resnet50-worker-2     Running 10s false   1        cn-shanghai.192.168.16.173
 pytorch-4-resnet50-worker-3     Running 10s false   1        cn-shanghai.192.168.16.173
```

3. Run the following command to print the job log:

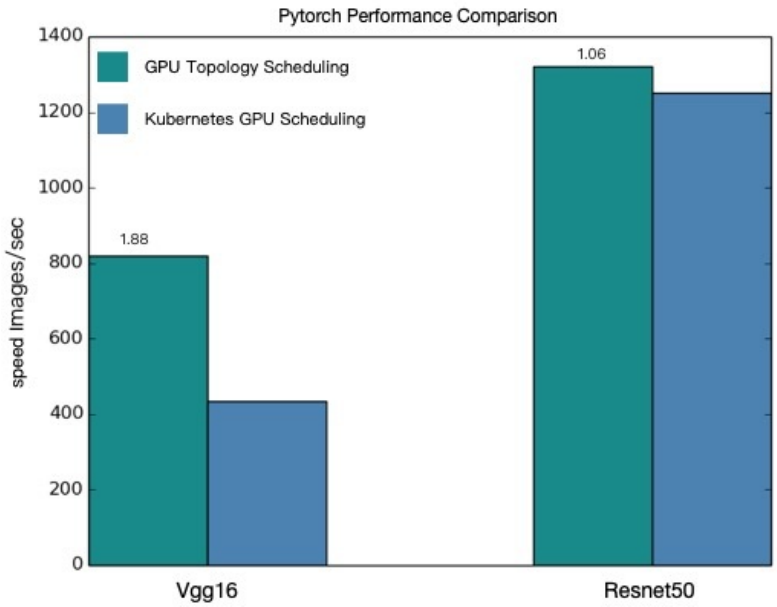
```
arena logs -f pytorch-4-resnet50
```

Expected output:

```
Model: resnet50
Batch size: 64
Number of GPUs: 4
Running warmup...
Running benchmark...
Iter #0: 313.1 img/sec per GPU
Iter #1: 312.8 img/sec per GPU
Iter #2: 313.0 img/sec per GPU
Iter #3: 312.2 img/sec per GPU
Iter #4: 313.7 img/sec per GPU
Iter #5: 313.2 img/sec per GPU
Iter #6: 313.6 img/sec per GPU
Iter #7: 313.0 img/sec per GPU
Iter #8: 311.3 img/sec per GPU
Iter #9: 313.6 img/sec per GPU
Img/sec per GPU: 313.0 +-1.3
Total img/sec on 4 GPU(s): 1251.8 +-5.3
```

Performance comparison

The following figure shows the performance comparison between GPU topology-aware scheduling and common GPU scheduling based on the preceding examples.



The figure shows that after GPU topology-aware scheduling is activated, the PyTorch distributed jobs are significantly accelerated.

> **Note** The improvement that is achieved by GPU topology-aware scheduling varies based on the models that you use and the cluster environment. You can evaluate the performance of your models based on the preceding examples.

Related information

- Overview
- Install the ack-ai-installer component

4.5. Task scheduling

This topic describes the gang scheduling and capacity scheduling features. Gang scheduling is suitable for tasks that require all-or-nothing scheduling. Capacity scheduling improves resource utilization based on elastic quotas.

Gang Scheduling

Gang scheduling is a scheduling algorithm that schedules all correlated processes to different processors in a parallel system and starts these processes at the same time. This prevents the process group from being blocked when the system fails to start some processes. For example, if you submit a batch job that contains multiple tasks, either all of the tasks are scheduled or none of them is scheduled. Task scheduling in the all-or-nothing scenario is known as gang scheduling. For more information about how to use gang scheduling, see Gang scheduling.

Capacity Scheduling

Kubernetes uses the ResourceQuota object to allocate resources statically. As a result, the resource utilization is not ideal. To improve the resource utilization of a Kubernetes cluster, Alibaba Cloud provides the capacity scheduling feature to optimize resource allocation. This feature is developed based on the Yarn capacity scheduler and the Kubernetes scheduling framework. This feature allows you to meet the resource requests in a Kubernetes cluster and improve resource utilization by sharing idle resources. You can use the capacity scheduling feature based on elastic quotas. For more information about how to use capacity scheduling, see Capacity Scheduling.

4.6. Use KMS to encrypt Kubernetes secrets at rest in the etcd

This topic describes how to use keys created in Key Management Service (KMS) to encrypt secrets in professional managed clusters of Alibaba Cloud Container Service for Kubernetes (ACK).

Prerequisites

- A customer master key (CMK) is created in the KMS console. For more information, see Manage CMKs.

  > **Note** Professional managed ACK clusters support only CMKs of the Aliyun_AES_256 type and do not support automatic rotation of CMKs.

- The ACK service account is authorized to assume the AliyunCSManagedSecurityRole role. If you use an unauthorized account to enable Kubernetes secret encryption at rest for a new or existing professional managed ACK cluster, you are prompted to authorize the account first.
- If the current account is a RAM user, make sure it has the AliyunKMSCryptoAdminAccess permission. For more information, see Grant permissions to a RAM user.

Context

Kubernetes secrets are used to store and manage sensitive data, such as passwords to applications, Transport Layer Security (TLS) certificates, and credentials to download Docker images. Kubernetes stores secrets in the etcd of the cluster.

You can use keys created in Key Management Service (KMS) to encrypt secrets in professional managed ACK clusters. Based on the envelope encryption mechanism, KMS automatically encrypts and decrypts secrets in the etcd of ACK clusters by using a KMS provider provided by Kubernetes. For more information about envelope encryption, see What is envelope encryption?. The following list describes the process to encrypt and decrypt secrets in a professional managed ACK cluster:

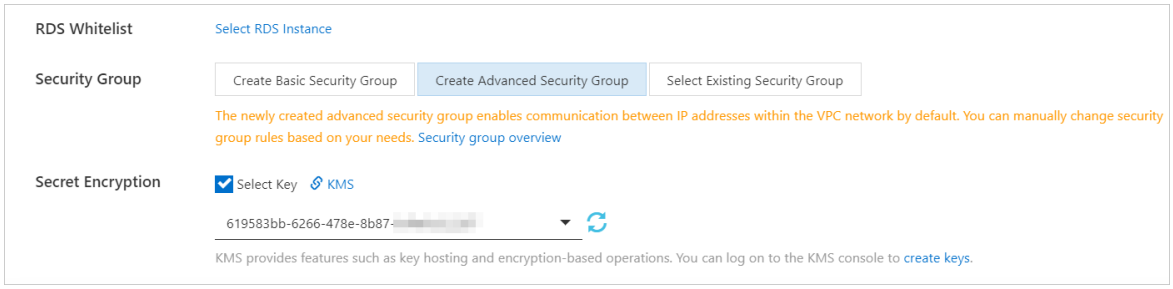
- When you use a Kubernetes secret to encrypt and store a password, the API server generates a random

data encryption key (DEK) to encrypt the secret. Then, the API server returns the DEK to KMS. KMS uses the specified key to encrypt the DEK and returns the encrypted DEK to the API server. The API server then stores the encrypted secret and DEK in etcd.

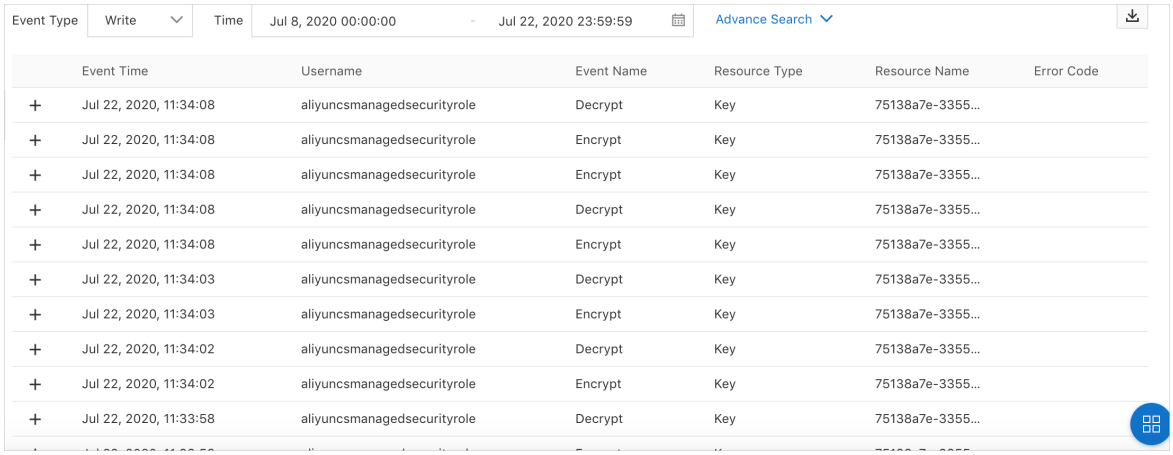
- When you decrypt the Kubernetes secret, the system calls the Decrypt API operation of KMS to decrypt the DEK. Then, the system uses the decrypted DEK to decrypt the Kubernetes secret and returns the password.

Enable Kubernetes secret encryption at rest when you create a professional managed ACK cluster

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the page, click **Create Kubernetes Cluster**. On the **Select Cluster Template** page, select **Professional Managed Cluster (Preview)** and click **Create**.
4. On the **ACK managed edition** tab, find the **Secret Encryption** section and select the **Select Key** check box. Then, select a CMK from the drop-down list. For more information about how to configure a professional managed ACK cluster, see Create a professional managed Kubernetes cluster.



Log on to the ActionTrail console. In the left-side navigation pane, click **History Search**. On the **History Search** page, check for encryption or decryption operations that are performed by using the AliyunKMSCryptoAdminAccess role. If these operations exist, the secret encryption at rest feature is enabled.



Enable secret encryption at rest for an existing professional managed ACK cluster

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, click the name of the cluster for which you want to enable secret encryption.
4. Click the **Basic Information** tab. In the **Basic Information** section, turn on the **Secret Encryption**

switch.

> **Note** If you are using a RAM user, make sure that the RAM user is granted the cluster administrator or O&M engineer role based on role-based access control (RBAC). For more information, see Assign RBAC roles to a RAM user.

If the status of the cluster changes from **Updating** to **Running**, the secret encryption at rest feature is enabled for the cluster.

Disable secret encryption at rest for an existing professional managed ACK cluster

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, click the name of the professional managed ACK cluster for which you want to disable secret encryption.
4. Click the **Basic Information** tab. In the **Basic Information** section, turn off the **Secret Encryption** switch.

> **Note** If you are using a RAM user, make sure that the RAM user is granted the cluster administrator or O&M engineer role based on role-based access control (RBAC). For more information, see Assign RBAC roles to a RAM user.

If the status of the cluster changes from **Updating** to **Running**, the secret encryption at rest feature is disabled for the cluster.

4.7. Customize the settings of control plane components in professional managed Kubernetes clusters

You can customize the settings of control plane components in a professional managed Kubernetes cluster to meet production needs. You can customize the settings of managed components such as Kube API Server and Kube Controller Manager (KCM). This topic describes how to customize the settings of control plane components in professional managed Kubernetes clusters.

Considerations

Before you customize the settings of a control plane component, take note of the following items:

- After you customize the settings of a component, the component is automatically restarted. We recommend that you customize the settings during off-peak hours.
- After you customize the settings, the changes overwrite the default settings of the professional managed Kubernetes cluster.
- To ensure the stability of the control plane component, you are allowed to customize only some of the settings.
- Make sure that the values of the customized parameters are valid and complete. Otherwise, the component may fail to be restarted. For more information about the parameters, see kube-apiserver and kube-controller-manager.

Customize the settings of a control plane component in a professional managed Kubernetes cluster

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.

   The following example shows how to customize the settings of Kube API Server:
5. In the **Core Components** section, find the component and click the icon.
6. In the **kube-apiserver Parameters** dialog box, set the parameters and click **OK**.

> **Note** Make sure that the specified values are valid and complete. You can customize only the settings of Kube API Server and KCM in professional managed Kubernetes clusters. For more information about the valid format and values of component parameters, see kube-apiserver and kube-controller-manager. Select the Kubernetes version based on the practical situation.

Default settings

The default settings are overwritten after you customize values for component parameters. You can reset the parameters to the default settings in the following table as needed.

Kubernetes version	Component	Parameter	Default value
1.16	kube-apiserver	ServiceNodePortRange	30000-32767
		EnableAdmissionPlugins	• If PodSecurityPolicy is enabled, the default value is `NodeRestriction,PodSecurityPolicy`. • If PodSecurityPolicy is disabled, the default value is `NodeRestriction`.
	kube-controller-manager	HorizontalPodAutoscalerSyncPeriod	15s
1.18	kube-apiserver	ServiceNodePortRange	30000-32767
		EnableAdmissionPlugins	• If PodSecurityPolicy is enabled, the default value is `NodeRestriction,PodSecurityPolicy`. • If PodSecurityPolicy is disabled, the default value is `NodeRestriction`.
	kube-controller-manager	HorizontalPodAutoscalerSyncPeriod	15s

5.Node management

5.1. Node

5.1.1. Add existing ECS instances to an ACK cluster

You can add existing Elastic Compute Service (ECS) instances to a Container Service for Kubernetes (ACK) cluster in the ACK console. ECS instances can be added to an ACK cluster only as worker nodes. This topic describes how to manually and automatically add ECS instances to an ACK cluster.

Prerequisites

- 创建Kubernetes托管版集群.
- The ECS instances that you want to add belong to the security group of the worker nodes in the ACK cluster. The security group is automatically created when you initialize the cluster.

> **Note**
> Before you use the cluster, take note of the following limits:
> - By default, you can deploy at most 100 nodes in each cluster. To add more nodes, Submit a ticket.
> - The ECS instances that you want to add to the ACK cluster must be deployed in the same region and virtual private cloud (VPC) as the cluster.
> - The ECS instances that you want to add must belong to the same account as the cluster.
> - Nodes that run the following operating systems can be added to an ACK cluster:
>   - Alibaba Cloud Linux 2.
>   - CentOS 7.x. CentOS 8.x and later are not supported.
>   - Windows Server 2019 and Windows Server Core, 1909.

Automatically add ECS instances

In auto mode, all ECS instances that are available within your account are listed. You can select, configure, and add one or more ECS instances to a cluster in the ACK console. After you complete the configurations, the ECS instances are automatically added to the cluster.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. Go to the **Add Existing ECS Instance** page in the following ways:
   - Find the cluster that you want to manage and click **More** in the Actions column to go to the **Add Existing ECS Instance** page.
     a. On the **Clusters** page, find the cluster that you want to manage and click **More** in the **Actions** column.
     b. Click **Add Existing Node** from the **More** drop-down list.
   - Find the cluster that you want to manage and click **Node Pools** in the Actions column to go to the **Add Existing ECS Instance** page.
     a. On the **Clusters** page, find the cluster that you want to manage and click **Node Pools** in the **Actions** column.
     b. On the **Node Pools** page, click **Add Existing Node** in the **Actions** column.

- Find the cluster that you want to manage and click **Applications** in the Actions column to go to the **Add Existing ECS Instance** page.
  a. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Applications** in the **Actions** column.
  b. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
  c. On the **Nodes** page, click **Add Existing Node**.

4. On the **Add Existing ECS Instance** wizard page, select the auto mode to automatically add ECS instances to the cluster.

   Set **Mode** to **Auto** and select the ECS instances that you want to add in the Select Existing ECS Instance section.

5. Click **Next Step** and configure the parameters on the **Specify Instance Information** wizard page.

Parameter	Description
Container Runtime	Select a type of container runtime for the nodes. You can select Docker or Sandboxed-Container.
System Images	Select an operating system for the nodes.
Custom Image	You can select a custom image to replace the default image.
Data Disk	Specify whether to store the container and image data on a data disk. ○ If the ECS instances have data disks mounted and the file system of the last data disk is not initialized, the system automatically formats the data disk to ext4. Then, the system mounts the data disk to /var/lib/docker and /var/lib/kubelet. **Note** After the data disk is formatted, the data that is stored on the disk is erased. Make sure that you have backed up the data before you add the ECS instances to the ACK cluster. ○ The system does not create and mount a new data disk if no data disk has been mounted to the ECS instances.
CPU Policy	Specify the CPU policy. Valid values: ○ none: indicates that the default CPU affinity is used. This is the default policy. ○ static: allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Logon Type	○ Key Pair: Select a key pair as the credential that is used to log on to the nodes. ○ Password: Enter and confirm the password that is used to log on to the nodes.
RDS Whitelist	Select a Relational Database Service (RDS) instance to add the ECS instances to the whitelist of the RDS instance.

Parameter	Description
Labels	Add labels to the nodes in the cluster. Enter one or more key-value pairs, and click **Add**. **Note** ○ Key is required. *Value* is optional. ○ Keys cannot start with aliyun, http://, or https://. Keys are not case-sensitive and cannot exceed 64 characters in length. ○ *Values* are not case-sensitive. A value must not exceed 128 characters in length, and cannot start with http:// or https://. ○ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the one that uses the same key. ○ You can add at most 20 labels to each resource. If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Retain Instance Name	By default, **Retain Instance Name** is turned on. If you do not want to retain the instance name, you can turn off **Retain Instance Name**. After you disable this feature, the node is renamed based on the node naming rules.
User Data	For more information, see Overview of ECS instance user data.

6. Click **Next Step**. In the message that appears, click **Confirm**.

Manually add ECS instances

Notice ECS instances that are manually added to an ACK cluster are not released when the ACK cluster is deleted.

In manual mode, you must obtain the installation command, log on to an ECS instance, and then run the command to add the ECS instance to an ACK cluster. You can add only one ECS instance at a time.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. Go to the **Add Existing ECS Instance** page in the following ways:
   - Find the cluster that you want to manage and click **More** in the Actions column to go to the **Add Existing ECS Instance** page.
     a. On the **Clusters** page, find the cluster that you want to manage and click **More** in the **Actions** column.
     b. Click **Add Existing Node** from the **More** drop-down list.
   - Find the cluster that you want to manage and click **Node Pools** in the Actions column to go to the **Add Existing ECS Instance** page.

a. On the **Clusters** page, find the cluster that you want to manage and click **Node Pools** in the **Actions** column.

b. On the **Node Pools** page, click **Add Existing Node** in the **Actions** column.

- Find the cluster that you want to manage and click **Applications** in the Actions column to go to the **Add Existing ECS Instance** page.

a. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Applications** in the **Actions** column.

b. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.

c. On the **Nodes** page, click **Add Existing Node**.

4. On the **Add Existing ECS Instance** wizard page, select the manual mode to add ECS instances to the cluster.

   Set **Mode** to **Manual** and select the ECS instances that you want to add in the Select Existing ECS Instance section.

5. Click **Next Step** and configure the parameters on the **Specify Instance Information** wizard page.

Parameter	Description
Data Disk	Specify whether to store the container and image data on a data disk. - If the ECS instances have data disks mounted and the file system of the last data disk is not initialized, the system automatically formats the data disk to ext4. Then, the system mounts the data disk to */var/lib/docker* and */var/lib/kubelet*. **Note** After the data disk is formatted, the data that is stored on the disk is erased. Make sure that you have backed up the data before you add the ECS instances to the ACK cluster. - The system does not create and mount a new data disk if no data disk has been mounted to the ECS instances.
RDS Whitelist	Select an RDS instance to add the ECS instances to the whitelist of the RDS instance.
Retain Instance Name	By default, **Retain Instance Name** is turned on. If you do not want to retain the instance name, you can turn off **Retain Instance Name**. After you disable this feature, the node is renamed based on the node naming rules.

6. Click **Next Step** to go to the **Complete** wizard page. On the **Complete** wizard page, copy the command and click **Complete**.
7. Log on to the ECS console. In the left-side navigation pane, click **Instances**, select the region where the ACK cluster is deployed, and then find the ECS instance that you want to add.
8. Click **Connect** in the Actions column. In the dialog box that appears, follow the instructions to connect to the ECS instance. Enter the remote connection password and click **OK**. After you are connected to the ECS instance, paste the command that is copied in Step 6 and click **OK** to execute the script. After the script is executed, the ECS instance is added to the cluster.

Result

On the **Clusters** page, find the cluster to which you added the ECS instance. Then, click the name of the cluster or click **Details** in the **Actions** column. In the left-side navigation pane of the details page, choose

Nodes > Nodes. On the **Nodes** page, you can view information about the newly added node.

5.1.2. Monitor nodes

Container Service for Kubernetes (ACK) clusters are integrated with the Cloud Monitor service. You can view monitoring data of nodes in ACK clusters. Metrics are collected from Elastic Compute Service (ECS) instances where the nodes run.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the **Nodes** page, find the node that you want to manage and click **Monitor** next to the node.
6. You are then redirected to the Cloud Monitor console. On the **Host Monitoring** page, click **Basic Monitoring** to view the basic information of the ECS instance, such as the CPU usage, inbound bandwidth (from Internet to VPC), outbound bandwidth (from VPC to Internet), average disk blocks per second (BPS), and average disk input/output operations per second (IOPS).

What's next

- To view operating system metrics, you must first install the Cloud Monitor agent. For more information, see Overview.
- To view process metrics, you must first install the Cloud Monitor agent. For more information, see Process monitoring.
- The monitoring feature is available for application groups in ACK clusters. For more information, see Monitor basic resources.

5.1.3. Manage node labels

You can manage node labels in the Container Service for Kubernetes (ACK) console. You can add a label to multiple nodes at a time, filter nodes by label, and delete labels.

Prerequisites

An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.

Context

You can use labels to schedule nodes. For more information, see Mark a node as unschedulable.

Add a label to multiple nodes at a time

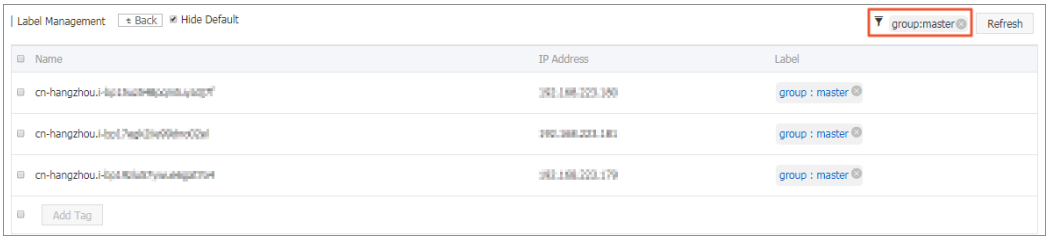
1. Log on to the Container Service for Kubernetes (ACK) console.
2. Go to the **Manage Labels and Taints** page.
   i. In the left-side navigation pane of the ACK console, click **Clusters** to go to the Clusters page.
   ii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
   iii. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
   iv. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner of the page.
3. Click the **Labels** tab. Select multiple nodes and click **Add Label**.
4. In the **Add** dialog box, set **Name** and **Value**, and then click **OK**.

On the **Labels** tab, you can verify that the label is added to the selected nodes.

Filter nodes by label

1. Log on to the ACK console.
2. Go to the **Manage Labels and Taints** page.
   i. In the left-side navigation pane of the ACK console, click **Clusters** to go to the Clusters page.
   ii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
   iii. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
   iv. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner of the page.
3. Click the **Labels** tab and click a label in the Labels column to filter the nodes.

   The page automatically refreshes and displays the nodes that have the specified label.



Delete a label

1. Log on to the ACK console.
2. Go to the **Manage Labels and Taints** page.
   i. In the left-side navigation pane of the ACK console, click **Clusters** to go to the Clusters page.
   ii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
   iii. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
   iv. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner of the page.
3. Click the **Labels** tab, select a node, and then click the Delete icon of a label in the Labels column to delete the label.
   After the label is deleted, it disappears from the Labels column.

5.1.4. Mark a node as unschedulable

This topic describes how to mark a node as schedulable or unschedulable in the Container Service for Kubernetes (ACK) console. This allows you to optimize the distribution of the loads on each node.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4.
5. On the **Nodes** page, select the node you want to manage and click **Set to Unschedulable**.
6. In the **Set to Unschedulable** dialog box, set the scheduling policy.

- **Set to Unschedulable**: Pods will not be scheduled to this node when you deploy new applications.
- **Set to Unschedulable and Drain**: Pods will not be scheduled to this node when you deploy new applications. Pods on this node will be evicted, except for the pods that are managed by DaemonSet.

7. Click **OK**.

   The status of the specified node is changed to Unschedulable.

What's next

To mark a node as schedulable, select the node and click **Set to Schedulable**. In the message that appears, click **OK**. The status of the specified node is changed to **Schedulable**.

5.1.5. Manage nodes in batches

You can use Operation Orchestration Service (OOS) to manage cluster nodes in batches. This allows you to enhance operations efficiency. For example, you can obtain logs of multiple nodes at the same time. This topic describes how to manage nodes in batches.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a managed Kubernetes cluster.
- OOS is activated in the OOS console.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the left-side navigation pane, choose **Clusters > Nodes**.
4. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
5.
6. Select the cluster and the nodes that you want to manage and click **Set to Unschedulable**.
7. In the **Batch Operations** dialog box, select the operations and maintenance (O&M) task that you want to run and click **OK**.
8. Set the parameters to run OOS tasks.

   For more information about the parameters, see .
9. After you set the parameters, click **Create**.

   You are redirected to the **Executions** page in the OOS console by default. You can find the newly submitted execution.

   - If you set Execution Mode to **Automatic**, the execution automatically starts. After the execution is complete, check the execution status.
   - If you set Execution Mode to **Manual**, you need to manually start the execution.

   To view the execution status, click the ID of the execution that you want to view or click **Details** in the **Actions** column.

5.1.6. Upgrade the configurations of a master node

This topic describes how to upgrade the configurations of a master node. You are responsible for the management of master nodes in a dedicated Kubernetes cluster. When the cluster size grows, you may need to upgrade the configurations of master nodes.

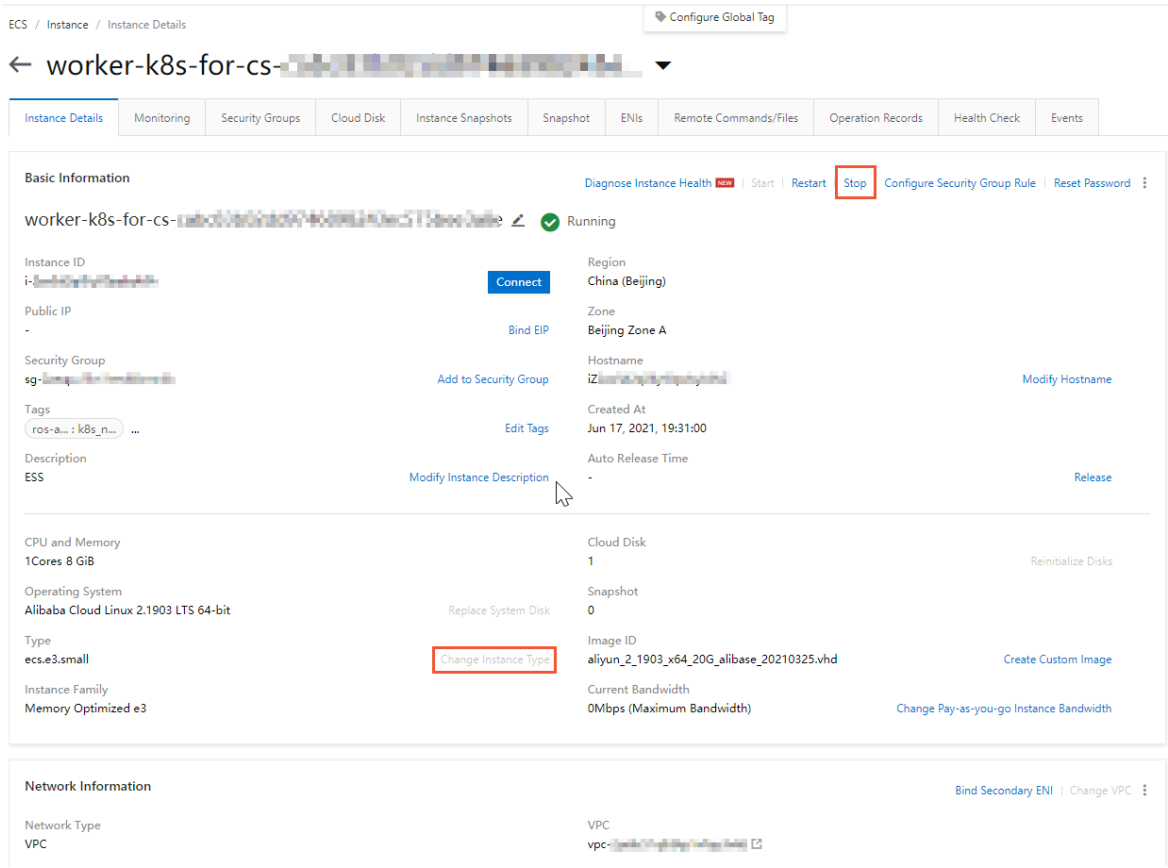
Prerequisites

Create a dedicated Kubernetes cluster

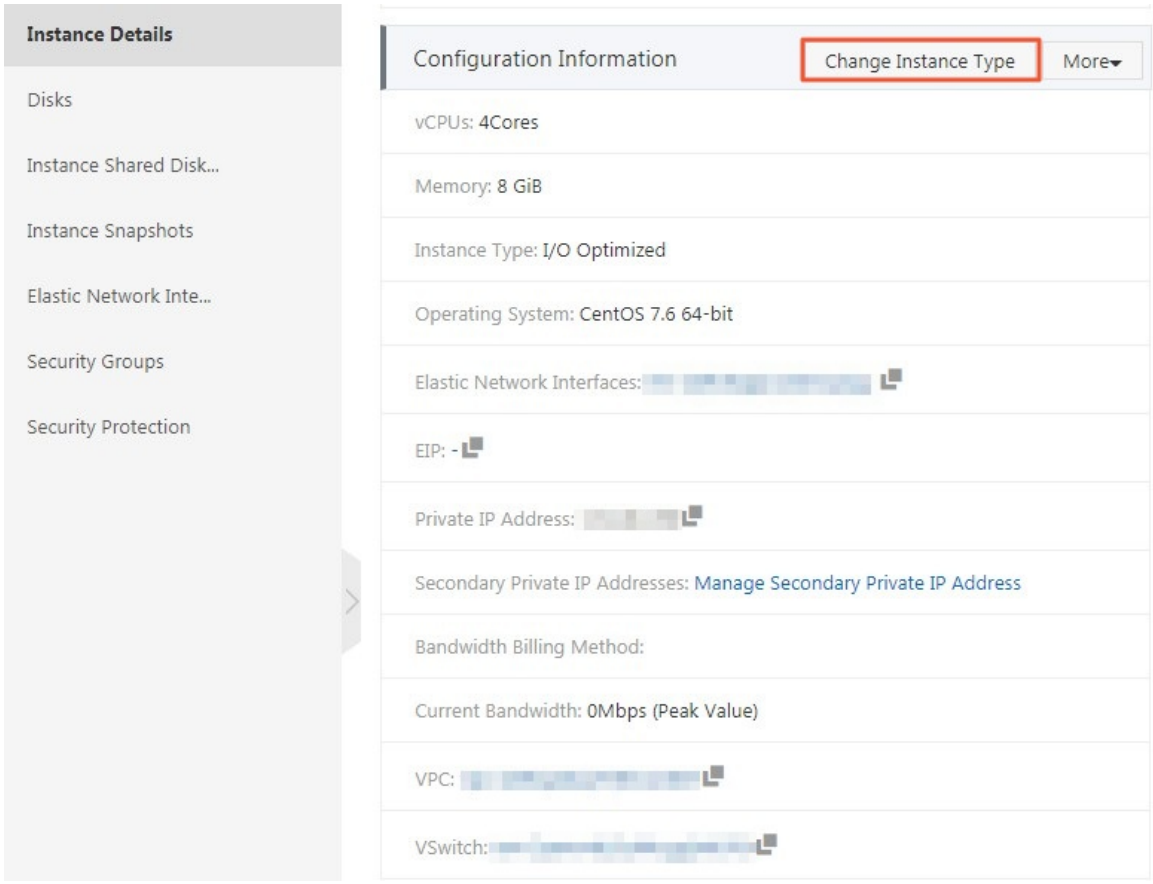
Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. Find the three master nodes of the cluster and click the instance ID of a master node. In this example, master-01 is selected. On the **Instance Details** page,

   you can view instance information such as the instance type.



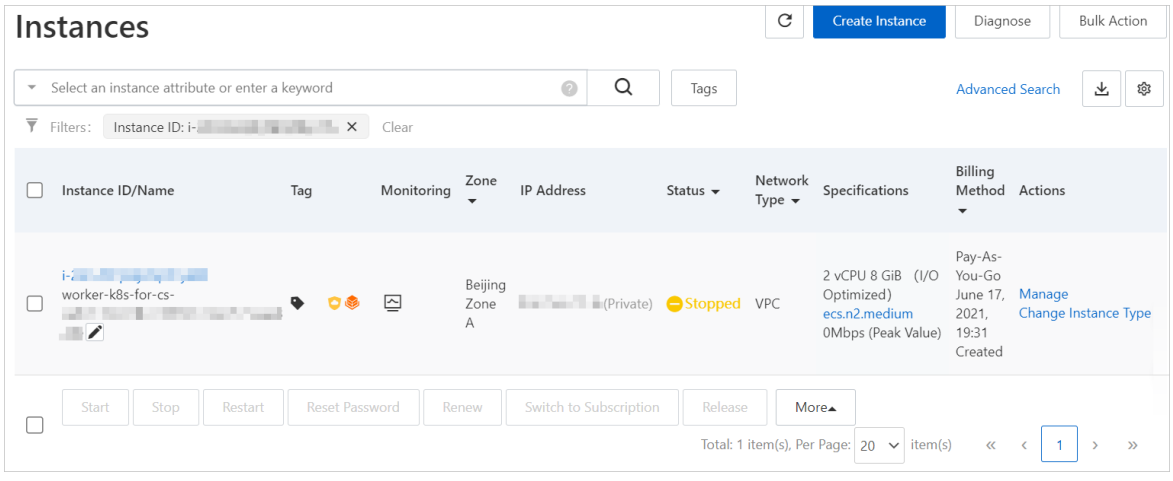
6. In the Configuration Information section, click **Change Instance Type**.



If the **Change Instance Type** button is dimmed, it indicates that the instance is running. Before you change the instance type, you must click **Stop** in the upper-right corner to stop the instance.

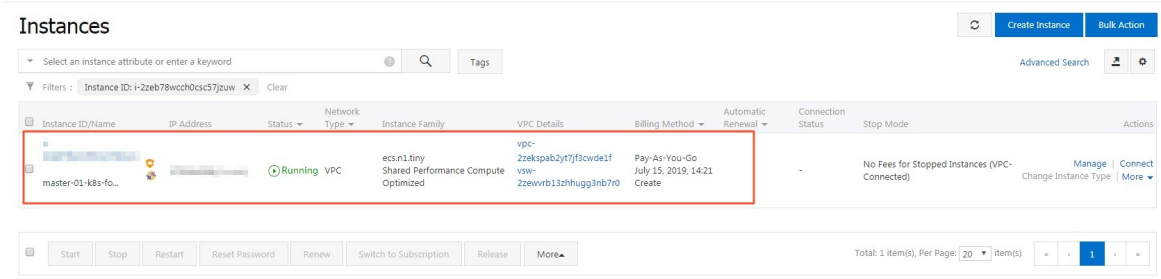
7. On the **Instance Type** page, select an instance type and click **Confirm**. For more information, see Select ECS instances to create the master nodes.
8. In the **Change success** message, click **OK**.

   Then, you are redirected to the **Instances** page.



The instance type of master-01 is changed.

9. Select the instance and click **Start**. Then, master-01 is automatically added to the cluster and the state of the node changes to **Running**, as shown in the following figure.



10. Repeat Step 4 to Step 7 to upgrade the configurations of the other master nodes.

> **Note** The master nodes in this topic are billed on a pay-as-you-go basis. For more information about how to upgrade the configurations of master nodes that are billed by other billing methods, see Overview of instance upgrade and downgrade.

5.1.7. Add worker nodes

If the workload of your cluster increases, the cluster resources may be insufficient. In this case, we recommend that you add resources to your cluster. For example, if your cluster contains more than 10 nodes, you can add worker nodes to improve resource utilization and reduce O&M complexity. This topic describes how to add worker nodes to a cluster.

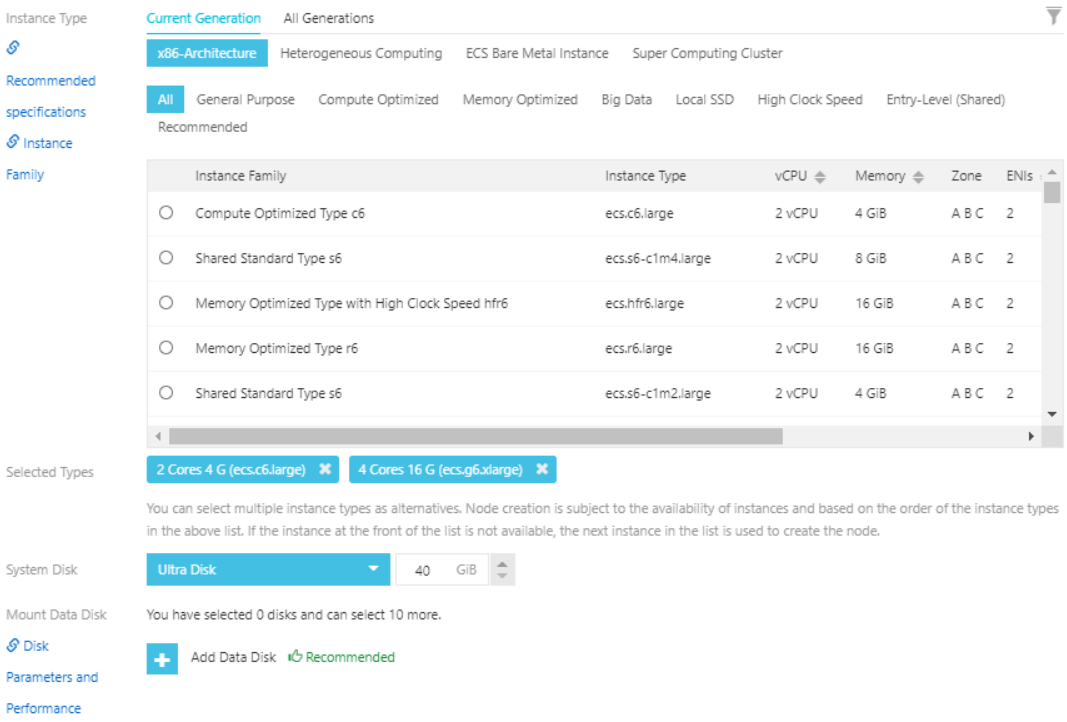
Prerequisites

A Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群.

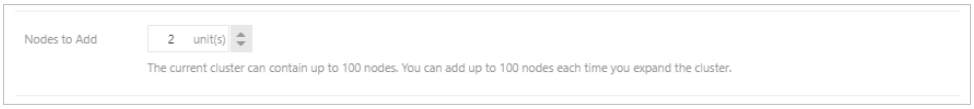
Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. Purchase new nodes or add existing nodes as needed.
   - Perform the following steps to purchase new nodes and add them to the cluster. For more information, see Expand an ACK cluster.
     a. Find the cluster to which you want to add nodes, and choose **More** > **Expand** in the **Actions** column.

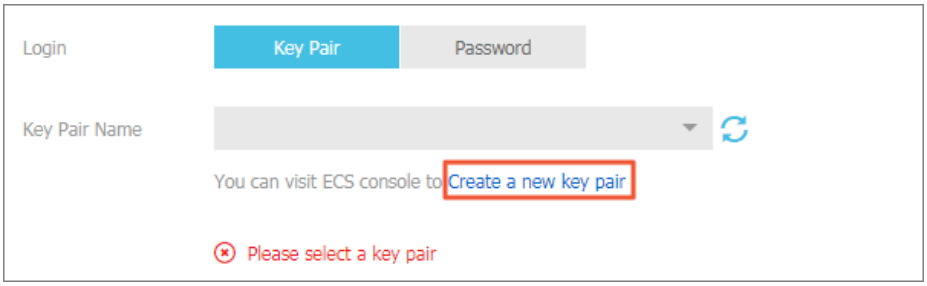
b. Configure the new worker nodes.



c. Set the number of worker nodes to add.



d. Set the logon type.



e. Set RDS Whitelist.

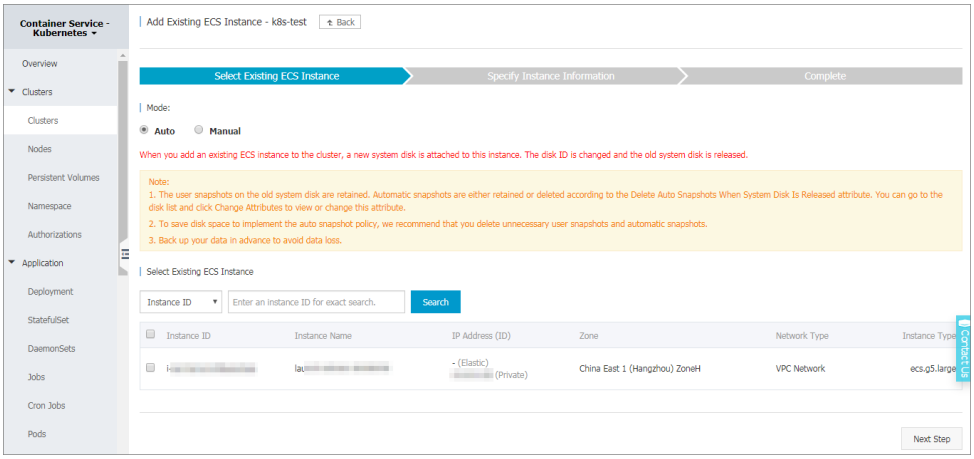
f. (Optional)Attach labels to new nodes.

g. After the configuration is complete, click **Submit**.

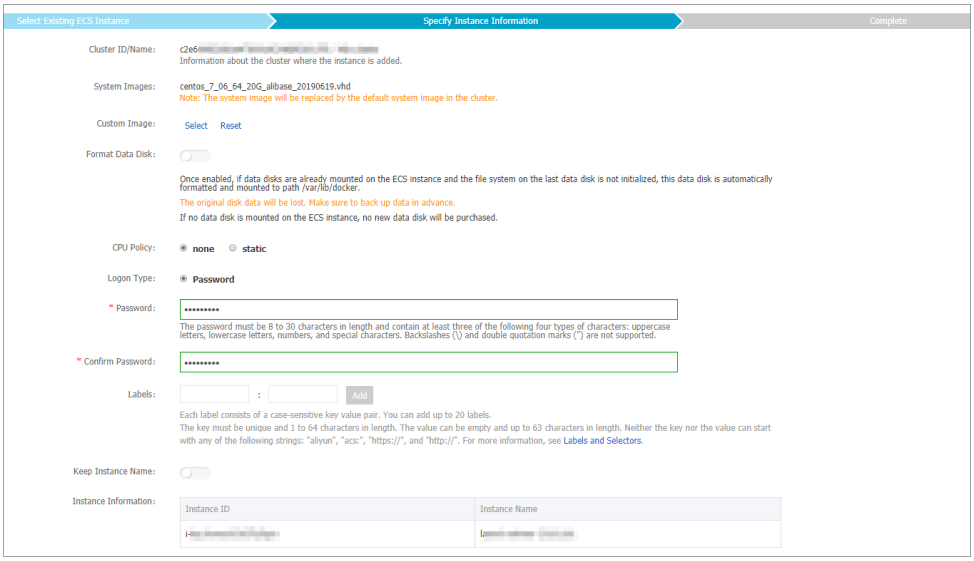
- To add existing nodes to a cluster, perform the following steps:

a. Find the cluster to which you want to add nodes, and choose **More > Add Existing Node** in the **Actions** column.
On the Add Existing ECS Instance page, you can select the **Auto** or **Manual** mode to add ECS instances. For more information, see Add existing ECS instances to an ACK cluster. In this example, the Auto mode is selected.

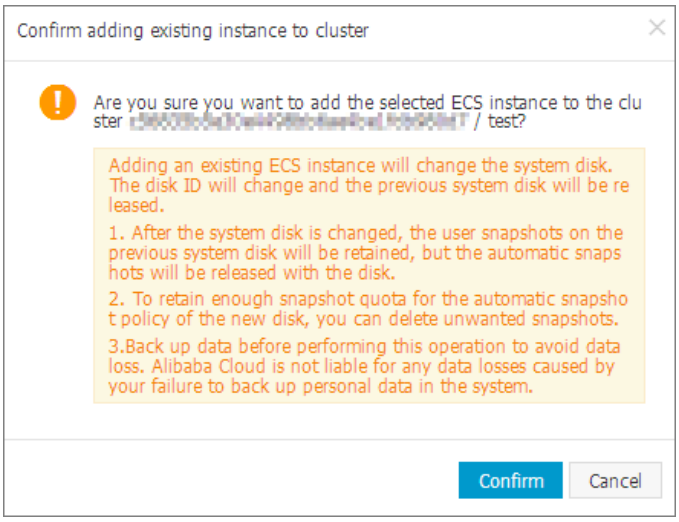
b. Set Mode to **Auto** and select instances from the ECS instance list. Then click **Next Step**.



c. Specify instance information, including the CPU policy, logon password, and labels, and click **Next Step**.



d. In the message that appears, click **Confirm**. The selected ECS instances are automatically added to the cluster.



4. On the management page of the cluster, click **Nodes** in the left-side navigation pane to view the status of the added nodes.

   If the added nodes are in the **Running** state, the operation is successful.

What's next

To remove a node from a cluster, find the node that you want to remove on the Nodes page, and choose **More > Remove** in the **Actions** column. In the message that appears, click **OK**.

> **Note**
>
> - To ensure the stability of your applications, we recommend that you add nodes to the cluster before you remove unnecessary nodes.
> - You can also configure PodDisruptionBudget (PDB) objects to ensure application stability when you remove nodes.

5.1.8. Mount a data disk to the Docker data directory

This topic describes how to mount a data disk to the Docker data directory. When the disk space is insufficient to support an increase in the number of containers or images on a node, you can mount a data disk to the Docker data directory. This provides more storage space to create containers and store images.

Docker data directory

Docker data is stored in a union file system (UnionFS) on a disk. The default container data and image data of Docker is stored in the */var/lib/docker* directory. You can run the **du** command to view the disk space that is occupied by this directory.

```
# du -h --max-depth=0 /var/lib/docker
7.9G   /var/lib/docker
```

Scenario

A Docker image may be large in size. A few images may occupy a large amount of disk space. The disk capacity may be insufficient if you create large-sized images or a large number of containers. In this case, you must mount a data disk to the Docker data directory. This allows you to create more images or containers.

Mount a data disk

To mount a disk to the Docker data directory, perform the following steps:

1. Create a data disk and mount it to the node for which you want to expand the disk. For more information, see Create a disk.
   i. Log on to the Elastic Compute Service (ECS) console and create the required disk.
   ii. In the left-side navigation pane, click **Instances**.
   iii. On the Instances page, click the ID of the ECS instance that you want to manage. Then, you are directed to the **Instance Details** page.
   iv. On the Instance Details page, click the **Cloud Disk** tab.
   v. On the Cloud Disk tab, click **Attach Disk** in the upper-right corner of the page.
   vi. In the dialog box that appears, select a disk from the **Target Disk** drop-down list. Click **OK**.
   vii. Click **Attach** to attach the disk to the specified ECS instance, and record the mount point `/dev/xvd*` or `/dev/vd*`.
2. Log on to the ECS instance and format the mounted disk.
   i. In the command-line interface (CLI), enter **ls -l /dev/xvd*** or **ls -l /dev/vd*** to check whether the returned mount point is the same as what you have recorded.
   ii. In the CLI, enter **fdisk** to partition the disk and enter **mkfs.ext4** to format the disk.

```
root@c836831d69e4040e797eff4d3c4dcd983-node2:~# ll /dev/xvd*
brw-rw---- 1 root disk 202,  0 May 26 15:44 /dev/xvda
brw-rw---- 1 root disk 202,  1 May 26 15:44 /dev/xvda1
brw-rw---- 1 root disk 202, 16 May 27 13:03 /dev/xvdb
root@c836831d69e4040e797eff4d3c4dcd983-node2:~# fdisk -S 56 /dev/xvdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x446953ae.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-62914559, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-62914559, default 62914559):
Using default value 62914559

Command (m for help): wq
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@c836831d69e4040e797eff4d3c4dcd983-node2:~# ll /dev/xvd*
brw-rw---- 1 root disk 202,  0 May 26 15:44 /dev/xvda
brw-rw---- 1 root disk 202,  1 May 26 15:44 /dev/xvda1
brw-rw---- 1 root disk 202, 16 May 27 13:08 /dev/xvdb
brw-rw---- 1 root disk 202, 17 May 27 13:08 /dev/xvdb1
root@c836831d69e4040e797eff4d3c4dcd983-node2:~# mkfs.ext4 /dev/xvdb1
mke2fs 1.42.9 (4-Feb-2014)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
1966080 inodes, 7864064 blocks
393203 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
240 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

3. Migrate the Docker data to the disk.

   If you do not want to suspend the applications that run on the node, you must migrate the applications. For more information about how to migrate applications that run in a Swarm cluster, see Schedule an application to specified nodes. For more information about how to migrate applications that run in a Kubernetes cluster, see Safely Drain a Node while Respecting Application SLOs.

   i. To ensure data integrity during the migration process, stop Docker daemon and kubelet. Enter **service docker stop** in the CLI to stop Docker daemon and enter **service kubelet stop** to stop kubelet.

   ii. Move the Docker data directory to a backup directory, such as **mv /var/lib/docker /var/lib/docker_data**.

iii. Mount the formatted disk to the */var/lib/docker* and */var/lib/kubelet* directories. Examples:

```
echo "/dev/xvdb1  /var/lib/container/  ext4  defaults    0 0" >>/etc/fstab
echo "/var/lib/container/kubelet /var/lib/kubelet none defaults,bind 0 0" >>/etc/fstab
echo "/var/lib/container/docker /var/lib/docker none defaults,bind 0 0" >>/etc/fstab
mkdir /var/lib/docker
mount -a
```

iv. Migrate the Docker data that has been backed up to the formatted disk, such as **mv /var/lib/docker_data/* /var/lib/docker/**.

4. Start Docker daemon and kubelet, and check where the data is stored.

i. In the CLI, enter **service docker start** to start Docker daemon and enter **service kubelet start** to start kubelet.

ii. In the CLI, enter **df** to check whether */var/lib/docker* has been mounted to the disk. If you want to start the Kubernetes cluster, skip this step.

```
root@c836831d69e4040e797eff4d3c4dcd983-node2:/var/lib# df
Filesystem     1K-blocks    Used Available Use% Mounted on
udev              497280       4    497276   1% /dev
tmpfs             101628     712    100916   1% /run
/dev/xvda1      41151808 1928420  37109960   5% /
none                   4       0         4   0% /sys/fs/cgroup
none                5120       0      5120   0% /run/lock
none              508136     288    507848   1% /run/shm
none              102400       0    102400   0% /run/user
/dev/xvdb1      30831612  667168  28575248   3% /var/lib/docker
```

iii. In the CLI, enter **docker ps** to check whether containers are lost. Restart containers as required. For example, you can restart containers that do not have the `restart:always` label.

```
root@c836831d69e4040e797eff4d3c4dcd983-node2:/var/lib# docker ps
CONTAINER ID        IMAGE                                                COMMAND                  CREATED             STATUS
4f564091bffa        registry.aliyuncs.com/acs/logspout:0.1-41e0e21       "/bin/logspout"          21 hours ago        Up 3 minutes
gspout_2
a5aba5fbedae        registry.aliyuncs.com/acs/ilogtail:0.9.9             "/bin/sh -c 'sh /usr/"   21 hours ago        Up 3 minutes
gtail_2
5e3d8fe154bb        registry.aliyuncs.com/acs/monitoring-agent:0.7-1cf85e6   "acs-mon-run.sh --hel"   21 hours ago        Up 3 minutes
_acs-monitoring-agent_1
fb72c2388b0e        registry.aliyuncs.com/acs/volume-driver:0.7-252cb09  "acs-agent volume_exe"   21 hours ago        Up 3 minutes
er_volumedriver_2
604fcb4ad720        registry.aliyuncs.com/acs/routing:0.7-c8c15f0        "/opt/run.sh"            21 hours ago        Up 3 minutes
uting_1
8fe1d6ed15b5        registry.aliyuncs.com/acs/agent:0.7-6967e86          "acs-agent join --nod"   21 hours ago        Up 3 minutes
999da3883264        registry.aliyuncs.com/acs/tunnel-agent:0.21          "/acs/agent -config=c"   21 hours ago        Up 3 minutes
```

5. If a container has been migrated to other nodes, you can schedule it back to the node to which you have attached the disk.

   For more information, see Container Service for Kubernetes.

5.1.9. Attach a data disk to a node

The existing disk capacity may be insufficient to support the increasing number of containers or images on a node in a Kubernetes cluster. You can attach one or more data disks to the node. This allows you to expand the disk capacity and provide more storage space for containers and images. This topic describes how to attach a data disk to a node.

Prerequisites

Kubernetes version 1.10.4 or later is used in the cluster.

Attach a data disk to the node

To expand the disk capacity of an existing node, you can use either of the following methods:

- If no disk is attached to the node, attach a data disk to the node. For more information, see Mount a disk to the Docker data directory.

- If you have purchased a data disk for the node, but the disk has failed to be attached to the node, perform the following steps:

> **Note**
> - To minimize the potential risks, we recommend that you create snapshots or backups of the node.
> - Make sure that the applications deployed on the node can be scheduled to other nodes in the cluster.
> - Perform the steps during off-peak hours.
> - If you drain the node, the pods on this node are scheduled to other nodes. Make sure that the cluster has sufficient nodes. If no sufficient nodes are available, we recommend that you temporarily scale out the nodes before the node is drained.

Before you attach the data disk, run the **df** command on the worker node. If the output indicates that `/var/lib/docker` is mounted to `/dev/vdb1`, the data disk is attached to the node. Then, you can skip the following steps. If the /var/lib/docker directory is not mounted, perform the following steps:

```
[root@[redacted]~]# df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/vda1       41151808 2273772  36764604   6% /
devtmpfs         3995592       0   3995592   0% /dev
tmpfs            4005096       0   4005096   0% /dev/shm
tmpfs            4005096     508   4004588   1% /run
tmpfs            4005096       0   4005096   0% /sys/fs/cgroup
/dev/vdb1      101441464   61668  96120584   1% /var/lib/docker
tmpfs             801020       0    801020   0% /run/user/0
```

1. Set the node to the unschedulable state.

   For more information, see Mark node as unschedulable.

2. Drain the node.

   For more information, see Safely-Drain-Node.

3. Remove the node.

   This topic only describes the operations in the Container Service for Kubernetes console.

   i. Log on to the ACK console.

   ii. In the left-side navigation pane of the ACK console, click **Clusters**.

   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster, or click **Applications** in the **Actions** column.

   iv. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.

   v. Select a node that you want to remove, and click **Batch Remove**, or choose **More > Remove** in the Actions column of the node.

   vi. In the **Remove Node** dialog box, click **OK**.

4. Add the removed node to the cluster.

   i. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.

   ii. On the Nodes page, click **Add Existing Node**.

   iii. On the Select Existing ECS Instance wizard page, set Mode to **Auto** or **Manual**. Auto is selected in this example.

iv. Select the Elastic Compute Service (ECS) that you want to add in the Select Existing ECS Instance section and click **Next Step**.

v. Follow the prompts as instructed to complete the required settings and click **Next Step**.

vi. In the **Confirm** message, click **OK**.

After the node is added to the cluster, log on to the node and run the **df** command to check whether the data disk is attached to the node.

The following figure shows that the data disk is attached to the node.

```
[root@                    ~]# df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/vda1       41151808 2273772  36764604   6% /
devtmpfs         3995592       0   3995592   0% /dev
tmpfs            4005096       0   4005096   0% /dev/shm
tmpfs            4005096     508   4004588   1% /run
tmpfs            4005096       0   4005096   0% /sys/fs/cgroup
/dev/vdb1      101441464   61668  96120584   1% /var/lib/docker
tmpfs             801020       0    801020   0% /run/user/0
```

This way, the data disk is attached to the existing node.

5.1.10. Manage taints

Pods repel nodes that have specified taints when pods are scheduled. You can add one or more taints to a node. This topic describes how to add a taint to multiple nodes at a time and delete a taint from a node.

Prerequisites

An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.

Add a taint to multiple nodes at a time

1. Log on to the ACK console.
2. Go to the **Manage Labels and Taints** page.
   i. In the left-side navigation pane of the ACK console, click **Clusters** to go to the Clusters page.
   ii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
   iii. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
   iv. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner of the page.
3. Click the **Taints** tab, select one or more nodes, and then click **Add Taint**.
4. In the dialog box that appears, set **Name**, **Value**, and **Effect**.
5. Click **OK**.
   On the **Taints** tab, you can verify that the taint is added to the selected nodes.

Filter nodes by taint

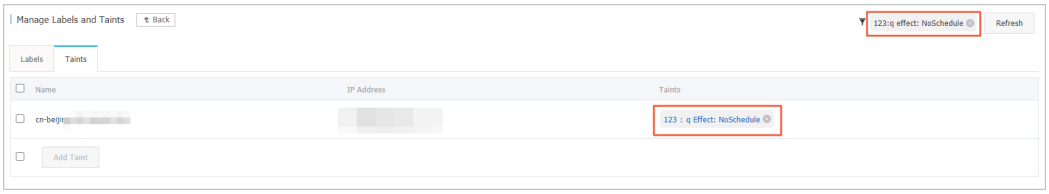
1. Log on to the ACK console.
2. Go to the **Manage Labels and Taints** page.
   i. In the left-side navigation pane of the ACK console, click **Clusters** to go to the Clusters page.
   ii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.

iii. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.

iv. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner of the page.

3. Click the **Taints** tab and click a taint in the Taints column to filter the nodes.

   The page automatically refreshes and displays the nodes that have the specified taint.



Delete a taint

1. Log on to the ACK console.
2. Go to the **Manage Labels and Taints** page.
   i. In the left-side navigation pane of the ACK console, click **Clusters** to go to the Clusters page.
   ii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
   iii. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
   iv. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner of the page.
3. On the **Manage Labels and Taints** page, click the **Taints** tab, select a node, and then click of a taint in the Taints column. In the message that appears, click **Confirm**.
   After the taint is deleted, it disappears from the Taints column.

5.1.11. Collect diagnostic logs

You can diagnose one or more nodes at a time in the Container Service for Kubernetes (ACK) console and collect the diagnostic logs. This topic describes how to collect diagnostic logs with simple steps.

Prerequisites

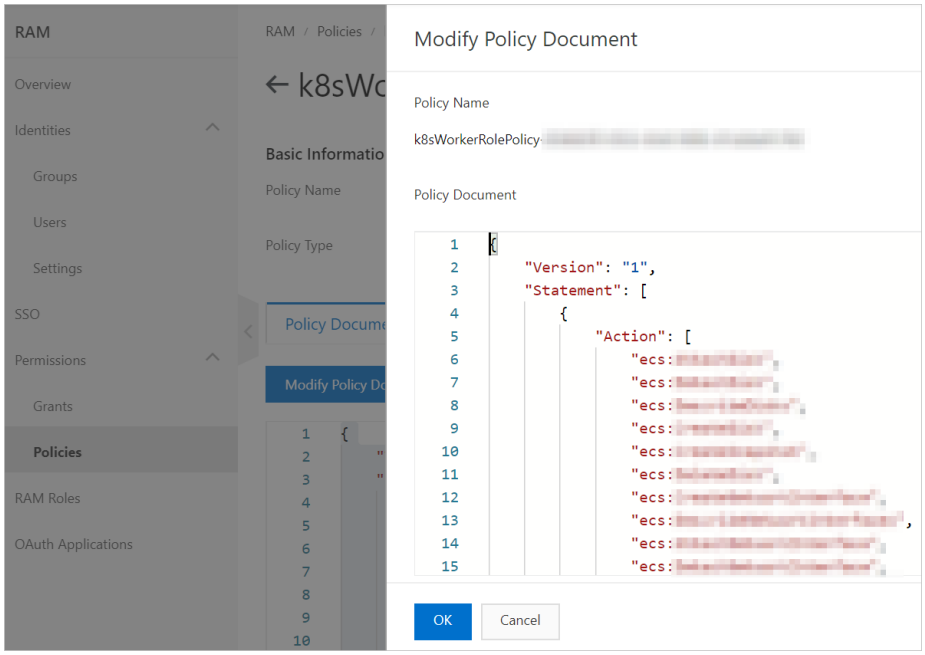
- 创建Kubernetes托管版集群
- Activate OSS

(Optional)

Authorize nodes to upload diagnostic logs to OSS

Before you can upload diagnostic logs from nodes to an Object Storage Service (OSS) bucket, you must grant the nodes the write permissions on the specified directory in OSS. Perform the following steps:

1. Log on to the RAM console by using your Alibaba Cloud account.
2. In the left-side navigation pane, choose **Permissions > Policies**.
3. On the **Policies** page, find the specific policy and click its name.
4. On the **Policy Document** tab, click **Modify Policy Document**.
5. In the **Modify Policy Document** pane, add the following content, and enter the bucket name and directory.




```
{
  "Action": [
    "oss:GetBucket",
    "oss:PutObject",
    "oss:GetObject"
  ],
  "Resource": [
    "acs:oss:*:*:<the OSS bucket name>/<the directory used to store diagnostic logs>/*"
  ],
  "Effect": "Allow"
},
{
  "Action": [
    "oss:GetBucketInfo"
  ],
  "Resource": [
    "acs:oss:*:*:<the OSS bucket name>"
  ],
  "Effect": "Allow"
}
```

6. Click **OK**.

Select nodes to diagnose

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. In the left-side navigation pane, choose **Clusters > Nodes**.
6. On the **Nodes** page, select one or more nodes, and click **Node Diagnosis**.
7. In the **Node Diagnosis** dialog box, perform the following operations:

- Select **Upload to OSS** and perform the following steps:
  You can upload the diagnostic logs to an OSS bucket. Before you perform the following steps, you must grant the nodes the write permissions on the specified directory in OSS. This allows the nodes to upload the node diagnostic logs to the specified OSS bucket. For more information, see Authorize nodes to upload diagnostic logs to OSS.
  a. Enter the name and directory of the OSS bucket.
  For example, if you want to upload diagnostic logs to */acs/diagnose* directory in the *myBucket*, enter *myBucket/acs/diagnose*.
  b. Select or clear **Share Diagnosis Logs**.
  If you select the check box, the system generates a temporary link for you to download the diagnostic logs. You can also share the link to the ACK support team to request technical support.
  c. Click **OK**.
  After the preceding steps are complete, you can obtain the **task ID** of Cloud Assistant in the console. Then, you can use the task ID to find and view the logs of diagnostic script executions in the Elastic Compute Service (ECS) Cloud Assistant console.
  d. In the **Node Diagnosis** dialog box, click **Go to Cloud Assistant to View Diagnosis Script Execution Logs**.
  e. On the **Cloud Assistant** page, click the **Command Execution Result** tab.
  f. Find the **Task ID** and click **View** in the Actions column.
  You can log on to the OSS console to view the diagnostic logs that are collected in the specified directory.
- Clear **Upload to OSS**, and click **OK**.
  If you clear **Upload to OSS**, you do not need to grant the nodes the write permissions on the specified directory in OSS. You can log on to ECS Console and choose **Maintenance and Monitoring > ECS Cloud Assistant > Command Execution Result** to view relevant diagnostic logs.

5.1.12. View node resource request rate and usage

You can view the resource usage on the nodes of a cluster in the Container Service for Kubernetes console.

Prerequisites

创建Kubernetes托管版集群

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. The Cluster Information page appears. In the left-side navigation pane, click **Nodes**.
   On the **Nodes** page, you can view the request rate and usage of CPU and memory on each node. The following formulas show how to calculate the request rate and usage:
   - CPU request rate = sum (The amount of CPU requested by all pods on the node)/Total CPU of the node
   - CPU usage rate = sum (The amount of CPU used by all pods on the node)/Total CPU of the node
   - Memory request rate = (The amount of memory requested by all pods on the node)/Total memory of

the node

- Memory usage rate = sum (The amount of memory used by all pods on the node)/Total memory of the node

> **Note**
>
> - You can adjust the workload of a node based on resource usage. For more information, see Mark a node as unschedulable.
> - If both the resource request and usage are 100% on a node, new pods will not be scheduled to this node.

5.1.13. View nodes

You can view nodes of a cluster by using the kubectl command-line interface (CLI) or the Container Service for Kubernetes console.

View nodes by using kubectl

> **Note** Before you run commands to view the nodes in a Kubernetes cluster, you must connect to the cluster by using kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

Use kubectl to connect to a cluster and run the following command to view the nodes in the cluster:

```
kubectl get nodes
```

The following example shows the output:

```
kubectl get nodes
NAME           STATUS   AGE    VERSION
iz2ze2n6ep53tch701y****  Ready   19m    v1.6.1-2+ed9e3d33a07093
iz2zeafr762wibijx39****  Ready   7m     v1.6.1-2+ed9e3d33a07093
iz2zeafr762wibijx39****  Ready   7m     v1.6.1-2+ed9e3d33a07093
iz2zef4dnn9nos8elyr****  Ready   14m    v1.6.1-2+ed9e3d33a07093
iz2zeitvvo8enoreufs****  Ready   11m    v1.6.1-2+ed9e3d33a07093
```

View nodes in the Container Service for Kubernetes console

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4.

5.1.14. Remove nodes from an ACK cluster

This topic describes how to remove nodes from a cluster of Container Service for Kubernetes (ACK).

Prerequisites

- Create a dedicated Kubernetes cluster
- Connect to Kubernetes clusters by using kubectl

Context

- When you remove nodes from an ACK cluster, you may need to migrate pods that run on these nodes to other nodes. This may cause service interruption. We recommend that you remove nodes during off-peak hours.
- Unknown errors may occur when you remove nodes. We recommend that you back up the data on these nodes before you remove the nodes.
- Nodes remain in the unschedulable state when they are being removed.
- Only worker nodes can be removed.
- We recommend that you remove nodes in the ACK console. If you run the **kubectl delete node** command to remove nodes from an ACK cluster, note the following limits:
  - The removed nodes cannot be added to other ACK clusters.
  - The Elastic Compute Service (ECS) instances are automatically released after the nodes are removed.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4.
5. On the **Nodes** page, find the node that you want to remove and choose **More > Remove** in the **Actions** column for the node.

   > **Note** To remove multiple nodes at a time, select the nodes that you want to remove on the **Nodes** page and click **Batch Remove**.

6. (Optional)In the **Remove Node** dialog box, select the **Release ECS Instance** and **Drain the Node** check boxes, and click **OK**.

   After the node is removed, the ECS instance is released. This is the ECS instance where the node is deployed.

   - **Release ECS Instance**:
     - Select this option to release only pay-as-you-go ECS instances.
     - Subscription ECS instances are automatically released after the subscription expires.
     - If you do not select the **Release ECS Instance** check box, the system continues to bill the ECS instance where the node is deployed.
   - **Drain the Node**: Select this option to migrate pods that run on the removed nodes to other nodes in the cluster. If you select this option, make sure that the other nodes in the cluster have sufficient resources for these pods.
     You can also run the `kubectl drain node-name` command to migrate pods that run on removed nodes to other nodes in the cluster.

     > **Note** The value of *node-name* must be in the format of *your-region-name.node-id*. For example, the value can be *cn-hangzhou.i-xxx*.
     >
     > - *your-region-name* specifies the region where the cluster is deployed.
     > - *node-id* specifies the ID of the ECS instance where the node is deployed.

5.2. Node pool management

5.2.1. Schedule an application to a specific node pool

Labels play an important role in Kubernetes. Services, deployments, and pods are associated by labels. You can set pod scheduling policies related to node labels to schedule pods to nodes that have specified labels. This topic describes how to schedule an application pod to a specified node pool.

Procedure

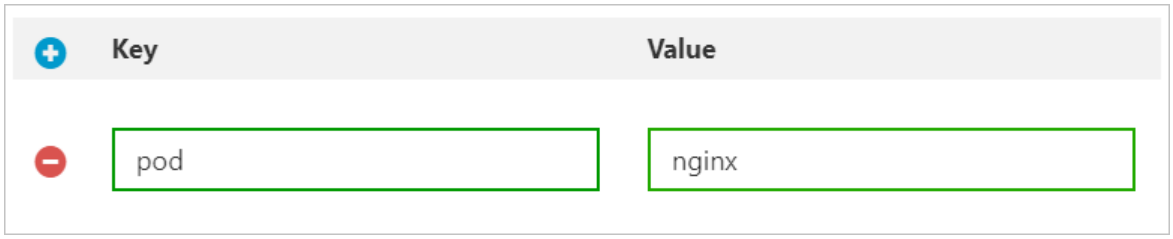
1. Specify a label for a node pool.

   In Alibaba Cloud Container Service for Kubernetes (ACK), you can manage a group of cluster nodes in a node pool. For example, you can manage labels and taints of all nodes in a node pool. For more information about how to create a node pool, see 管理节点池.

   i. Log on to the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the right side of the cluster that you want to manage, click **Node Pools**.
   iv. In the upper-right corner of the **Node Pools** page, click **Create Node Pool**.
   v. On the Create Node Pool page, click **Show Advanced Options**, and then click the icon to add **labels** to nodes.

      In this example, the label added to the node is pod: nginx.

   You can also click **Scale Out** on the right side of a node pool to update or add labels for nodes. If automatic scaling is enabled for a node pool, click **Modify** on the right side of the node pool to update or add labels for nodes.

2. Set a scheduling policy for the application pod.

   In the preceding step, the pod: nginx label is attached to the nodes in the node pool. You can use the nodeSelector or nodeAffinity field to schedule an application pod to a specified node pool. The following content describes how to schedule an application pod:

   - Set nodeSelector.
     nodeSelector is a field in the spec section. Add the pod: nginx label to nodeSelector. Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment-basic
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   nodeSelector:
    pod: nginx    # After you add the label of a node pool here, this application pod can run only on nodes i
n the node pool.
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```

- Set nodeAffinity.
  You can also use nodeAffinity to schedule application pods. The following four scheduling policies are available:

  ```
  - requiredDuringSchedulingIgnoredDuringExecution
  ```

  If this policy is used, the pod must be deployed on a node that meets the requirements. If no nodes meet the requirements, the system retries until a node that meets the requirements is found. IgnoreDuringExecution indicates that if the tag of the node where the pod is deployed changes and does not meet the requirements, the pod keeps running on the node.

  ```
  - requiredDuringSchedulingRequiredDuringExecution
  ```

  If this policy is used, the pod must be deployed on a node that meets the requirements. If no nodes meet the requirements, the system retries until a node that meets the requirements is found. RequiredDuringExecution indicates that if the tag of the node where the pod is deployed changes and does not meet the requirements, the system selects another node that meets the requirements to deploy the pod.

  ```
  - preferredDuringSchedulingIgnoredDuringExecution
  ```

  If this policy is used, the pod is preferentially deployed on a node that meets the requirements. If no nodes meet the requirements, the system ignores these requirements.

  ```
  - preferredDuringSchedulingRequiredDuringExecution
  ```

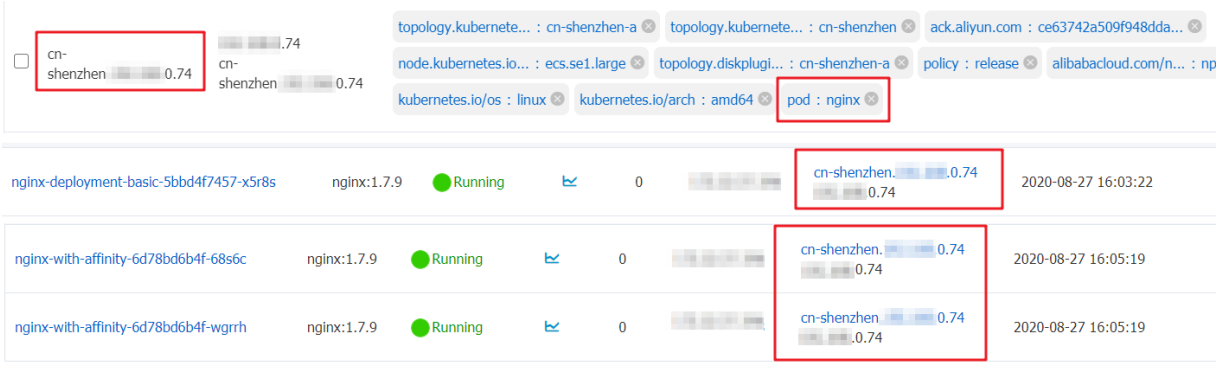
  If this policy is used, the pod is preferentially deployed on a node that meets the requirements. If no nodes meet the requirements, the system ignores these requirements. RequiredDuringExecution indicates that if the tag of a node changes and meets the requirements after the pod is deployed on another node, the system deploys the pod on the node that meets the requirements.

In the following example, the requiredDuringSchedulingIgnoredDuringExecution policy is used to ensure that the application runs on a node in the specified node pool.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-with-affinity
 labels:
  app: nginx-with-affinity
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx-with-affinity
 template:
  metadata:
   labels:
    app: nginx-with-affinity
  spec:
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
       - key: pod
         operator: In     # The application runs on a node that has the pod: nginx label.
         values:
         - nginx
   containers:
   - name: nginx-with-affinity
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```

Result

All application pods in the preceding examples are scheduled to the xxx.xxx.0.74 node, which is the node that has the pod: nignx label.



5.2.2. Set the ratio of preemptible instances to existing instances in a node pool

You can set the ratio of preemptible instances to existing instances in a node pool. This allows you to reduce costs by controlling the number of preemptible instances when the node pool contains sufficient existing instances. This topic describes how to set the ratio of preemptible instances to existing instances in a node pool for a Container Service for Kubernetes (ACK) cluster.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- The ACK cluster must run Kubernetes 1.9 or later.

> **Notice**
>
> - By default, you can deploy up to 100 nodes in each cluster. To increase the quota,Submit a ticket.
> - When you add an existing Elastic Compute Service (ECS) instance, make sure that the ECS instance is associated with an elastic IP address (EIP) or a NAT gateway is deployed for the virtual private cloud (VPC) to which the ECS instance belongs. In addition, you must make sure that the node has access to the Internet. Otherwise, you may fail to add the ECS instance.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. Select a node pool or create a node pool.
   - Find the node pool that you want to modify, and click **Edit** in the **Actions** column.
   - In the upper-right corner of the **Node Pools** page, click **Create Node Pool**. For more information, see 管理节点池.
6. In the **Create Node Pool** or **Edit Node Pool** dialog box, select multiple vSwitches for **VSwitch**, and select **Preemptible Instance** as **Billing Method**.
7. Click **Show Advanced Options**. Select **Cost Optimization** for **Multi-Zone Scaling Policy** and set the **Percentage of Pay-as-you-go Instances** and **Enable Supplemental Pay-as-you-go Instances** parameters to meet your business requirements.

   The following table describes the parameters.

Parameter	Description	Default

Parameter	Description	Default
Multi-zone Scaling Policy	○ Priority: Nodes are selected based on the order of instance types that you select. ○ Cost Optimization: Nodes are selected based on the costs of each instance types. The instance type with a lower cost is prioritized. ○ Distribution Balancing: Nodes are selected based on the node distribution across zones. This policy aims to evenly distribute nodes across multiple zones.	Distribution Balancing
Percentage of Pay-as-you-go Instances	The percentage of pay-as-you-go instances that are created in the node pool. This number does not include the guaranteed minimum number of ECS instances in the node pool.	20
Enable Supplemental Pay-as-you-go Instances	Specifies whether to automatically create pay-as-you-go instances to reach the expected number when preemptible instances are insufficient. For example, preemptible instances are out of stock or your bidding price is higher than the market price.	Enable

> **Note**
> After you click **Confirm Order**:
> - The value of **Multi-zone Scaling Policy** cannot be modified.
> - If you select **Cost Optimization** for **Multi-zone Scaling Policy**, you can modify the values of **Percentage of Pay-as-you-go Instances** and **Enable Supplemental Pay-as-you-go Instances**.

8. Click **Confirm Order**.

What's next

After you complete the configuration, you can find the node pool on the **Node Pools** page. Click **Details** in the **Actions** column to view **Percentage of Pay-as-you-go Instances** in the **Node Configurations** field.

5.3. Managed node pools

5.3.1. Overview

Managed node pools are provided by Container Service for Kubernetes (ACK). Managed node pools support auto upgrading and auto repairing. This provides centralized, managed, and operations and maintenance (O&M)-free lifecycle management on nodes. You do not need to be concerned about the O&M of cluster nodes, such as component upgrading, OS upgrading, and patching to fix Common Vulnerabilities and Exposures (CVE)-identified vulnerabilities. ACK automatically fixes node exceptions on nodes in a managed node pool.

Context

The following list describes the terms that are related to node pools:

- **Basic node pool**: You can use a basic node pool to manage a set of nodes that have the same configurations, such as specifications, labels, and taints. You can manually perform O&M operations on nodes in a basic node pool.
- **Managed node pool**: Managed node pools are based on basic node pools and provide additional features, such as auto upgrading and auto repairing on nodes. This provides automated and managed O&M on nodes.
- **Replace system disk**: You can initialize a node by replacing the system disk of the node. After the system disk is replaced, the attributes of the Infrastructure as a Service (IaaS) services that are attached to the node remain unchanged, such as the node name, the related instance ID, and the assigned IP addresses. However, the original data on the system disk is deleted. Then, the node is initialized again. Attached data disks are not affected.

  > **Notice** Do not use system disks to persist data. We recommend that you use data disks to persist data.

- **In-place upgrade**: You can directly replace components as required on the system disk of a node. This is an alternative to replacing the system disk of a node. In-place upgrades do not replace system disks, reinitialize nodes, or destroy the original data on nodes.

Scenarios

- Users focus on application development instead of the O&M of worker nodes.
- Users require quick patching to fix CVE-identified vulnerabilities. When a vulnerability is disclosed by CVE, upgrades are automatically performed on nodes to fix the vulnerability.
- Users require elasticity instead of immutability for workloads. The business pods of these users are insensitive to node changes and tolerable to migrations.
- Users require automatic upgrading of the Docker versions and operating system (OS) image versions on nodes.

Comparison between managed node pools and basic node pools

Item	Managed node pool	Basic node pool
O&M	Managed by ACK	Managed by users

Item	Managed node pool	Basic node pool
Node upgrading	• Auto upgrading during maintenance windows. • Manual upgrading is also supported to fix CVE-identified vulnerabilities or upgrade the OS versions, Docker versions, and kubelet versions. For more information, see Upgrade a managed node pool. • Upgrading is implemented by replacing system disks.	• Only manual upgrading is supported. • Only kubelet versions can be upgraded. • Upgrading is implemented by in-place upgrades.
Auto repairing	Supported	Not supported
Key management	Only key pairs are supported.	Key pairs and passwords are supported.

Features

- You can create multiple managed node pools in an ACK cluster. The configuration of each managed node pool can be different from the others. This allows you to manage multiple sets of nodes with different specifications for your cluster.
- Before a node is upgraded by replacing the system disk of the node, ACK runs the **kubectl cordon** command to change the node to the Unschedulable state. Then, pods on the node are evicted. If pods are not evicted within 15 minutes, ACK forcibly replaces the system disk.
- A managed node pool monitors the state of nodes in the node pool. If the state is not reported from a node for more than 10 minutes or a node is in the NotReady state, ACK restarts the node to restore the workloads on the node.
- You can also disable the auto upgrading feature for a managed node pool. After auto upgrading is disabled, ACK does not automatically upgrade the Docker versions or OS versions when later versions are released.
- You must set maintenance windows for auto upgrading on nodes. ACK performs upgrades during maintenance windows. If an upgrade is not completed at the end of a maintenance window, it is suspended and resumed at the beginning of the next maintenance window.

Prerequisites

- Managed node pools perform upgrades on nodes by replacing the system disks of nodes. This deletes the data on the previous system disks. Data disks are not affected. Do not use system disks to persist data.
- Before the system disk of a node is replaced, ACK disables and drains the node. This may restart pods and interrupt persistent connections.
- When exceptions occur on a node in a managed node pool, ACK may restart the node to fix the exceptions. This restarts pods on the node.

5.3.2. Manage managed node pools

Managed node pools support node auto-upgrade and auto-repair. This provides a centralized, managed, and operations and maintenance (O&M)-free lifecycle management of nodes. This topic describes how to create, scale out, clone, and delete a managed node pool.

Prerequisites

- A professional Kubernetes cluster is created. For more information, see Create a professional managed Kubernetes cluster.
- The Kubernetes version of the cluster must be later than V1.9.

Prerequisites

- Managed node pools perform upgrades on nodes by replacing the system disks of nodes. This deletes the data on the previous system disks. Data disks are not affected. Do not use system disks to persist data.
- Before the system disk of a node is replaced, ACK disables and drains the node. This may restart pods and interrupt persistent connections.
- When exceptions occur on a node in a managed node pool, ACK may restart the node to fix the exceptions. This restarts pods on the node.

Create a managed node pool

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. In the upper-right corner of the **Node Pools** page, click **Create Managed Node Pool**.
6. In the **Create Managed Node Pool** dialog box, set the parameters of the managed node pool.

   For more information about the parameters, see Create a professional managed Kubernetes cluster. The following list describes some of the parameters:

   - Maintenance Window: Set the time window for the auto-upgrade of nodes in the managed node pool. Click **Set** to go to the **Basic Information** tab of the details page of the cluster. In the Cluster Maintenance section, you can turn on **Maintenance Window** and modify the time window for maintenance.
   - Auto Scaling: Specify whether to enable auto scaling of the managed node pool. For more information, see Auto scaling of nodes.
   - Quantity: Specify the number of nodes that you want to add to the node pool. If you do not want to create nodes, set this parameter to 0.
   - Operating System: Select an operating system for the nodes. CentOS, Alibaba Cloud Linux, and Windows are supported.
   - Node Label: You can add labels to nodes.
   - ECS Label: You can add labels to ECS instances.
   - Custom Resource Group: You can specify the resource group to which the nodes in the node pool belong.
7. Click **Confirm Order**.
   On the **Node Pools** page, check the **Status** column of the managed node pool. If the node pool is in the **Initializing** state, it indicates that the node pool is being created. After the managed node pool is created, the managed node pool is in the **Active state**.

Upgrade a managed node pool

If the operating system of nodes in a managed node pool is upgradable, you can upgrade the node pool.

1. Find the managed node pool that you want to upgrade and click **Upgrade** in the **Actions** column.
2. In the **Upgrade Node Pool** dialog box, set the following parameters.

Parameter	Description
Required Version	By default, **OS Image** and **Container Runtime** are selected. ○ **OS Image**: After you select this check box, the operating system of all nodes is upgraded to the required version. ○ **Container Runtime**: After you select this check box, the container runtime of all nodes is upgraded to the required version.
Maximum Unschedulable Nodes	Specifies the maximum number of nodes that can be in the unschedulable state.
Additional Nodes	Specifies the number of additional nodes that are automatically added before the node pool is upgraded. These additional nodes are automatically removed after the node pool is upgraded. Additional nodes are used to host workloads from upgradable nodes when they are drained.

> **Note** The sum of the values of **Maximum Unschedulable Nodes** and **Additional Nodes** equals the number of nodes that can be concurrently upgraded.

Scale out a managed node pool

You can scale out a managed node pool by adding nodes with the same attributes as the existing nodes.

1. On the Node Pools page, find the node pool that you want to scale out and click **Scale Out** in the **Actions** column.
2. In the dialog box that appears, set **Nodes to Add**. You can select or enter the number of nodes to be added. Then, click **Submit**.

   > **Note** If you want to modify the configurations of the node pool, click **Modify Node Pool Settings**. For more information, see Create a managed node pool.

   On the **Node Pools** page, the node pool is in the **Scaling state** when it is being scaled. After the scaling process is complete, the **state** of the node pool is changed to **Active**.

Clone a managed node pool

You can create a node pool by cloning an existing node pool. The new node pool has the same configurations as the existing one.

1. On the Node Pools page, find the node pool that you want to clone and click **Clone** in the **Actions** column.
2. In the dialog box that appears, enter a name for the new node pool, set the parameters, and click **Submit**.

   On the **Node Pools** page, the node pool is in the **Scaling state** when it is being created. After the cloning process is complete, the **state** of the node pool is changed to **Active**.

Delete a managed node pool

Before you delete a managed node pool, we recommend that you remove nodes from the managed node pool to avoid service disruptions caused by user errors.

1. On the Node Pools page, find and click the node pool that you want to delete.
2. In the **Node Configurations** section, select all nodes on the **Nodes** tab and click **Remove Node**.
3. (Optional)In the **Remove Node** dialog box, select **Release ECS Instance** and **Drain the Node**, and

then click **OK**.

- **Release ECS Instance**:
  - This option releases only pay-as-you-go ECS instances.
  - Subscription ECS instances are automatically released after the subscription expires.
  - If you do not select **Release ECS Instance**, the system continues to bill the ECS instance where the node is deployed.
- **Drain the Node**: Select this option to migrate pods that run on the removed nodes to other nodes in the cluster. If you select this option, make sure that the other nodes in the cluster have sufficient resources for these pods.
  You can also run the `kubectl drain node-name` command to migrate pods that run on removed nodes to other nodes in the cluster.

  > **Note** The value of *node-name* must be in the format of *your-region-name.node-id*. For example, you can enter *cn-hangzhou.i-xxx*.
  >
  > - *your-region-name* specifies the name of the region where the cluster is deployed.
  > - *node-id* specifies the ID of the ECS instance where the node to be removed is deployed.

4. Go to the Node Pools page, find the node pool that you want to delete, and click **Delete** in the **Actions** column.
5. In the **Delete Node Pool** message, click **Confirm**.

Enable and disable managed node pools

You can enable managed node pools to convert default node pools, custom node pools, and auto-scaling node pools into managed node pools. You can also disable managed node pools. This section describes how to enable and disable managed node pools.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. Find the node pool that you want to manage and perform the following operations in the **Actions** column.
   - Click **Enable Managed Node Pool** to convert the node pool into a managed node pool.
   - Click **Disable Managed Node Pool** to convert the managed node pool into other types of node pools.
     After the managed node pool is disabled, the managed node pool is converted based on the original type of the node pool. If the original type is managed node pool, it is converted into a custom node pool. Otherwise, it is converted into the original type of the node pool. For example, a default node pool can be converted into a managed node pool. Then, if you disable the managed node pool, it is converted into a default node pool.

5.3.3. Schedule an application pod to a specified node pool

Labels play an important role in Kubernetes. Services, deployments, and pods are associated by labels. You can set pod scheduling policies related to node labels to schedule pods to nodes that have specified labels. This topic describes how to schedule an application pod to a specified node pool.

Procedure

1. Specify a label for a node pool.

   In Alibaba Cloud Container Service for Kubernetes (ACK), you can manage a group of cluster nodes in a node pool. For example, you can manage labels and taints of all nodes in a node pool. For more information about how to create a node pool, see 管理节点池.

   i. Log on to the ACK console.

   ii. In the left-side navigation pane of the ACK console, click **Clusters**.

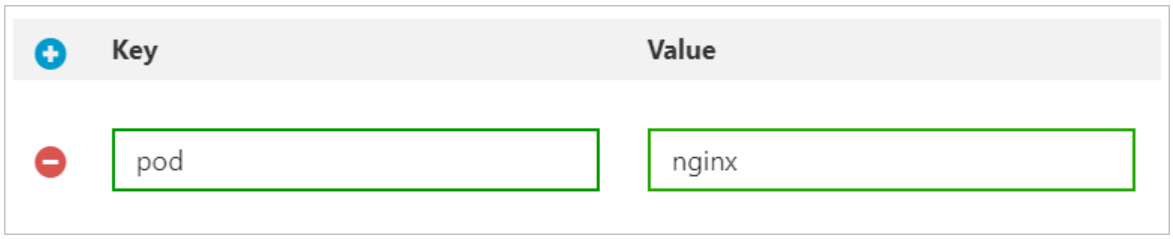
   iii. On the right side of the cluster that you want to manage, click **Node Pools**.

   iv. In the upper-right corner of the **Node Pools** page, click **Create Node Pool**.

   v. On the Create Node Pool page, click **Show Advanced Options**, and then click the icon to add **labels** to nodes.

   In this example, the label added to the node is pod: nginx.

   You can also click **Scale Out** on the right side of a node pool to update or add labels for nodes. If automatic scaling is enabled for a node pool, click **Modify** on the right side of the node pool to update or add labels for nodes.



2. Set a scheduling policy for the application pod.

   In the preceding step, the pod: nginx label is attached to the nodes in the node pool. You can use the nodeSelector or nodeAffinity field to schedule an application pod to a specified node pool. The following content describes how to schedule an application pod:

   - Set nodeSelector.
     nodeSelector is a field in the spec section. Add the pod: nginx label to nodeSelector. Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment-basic
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   nodeSelector:
    pod: nginx    # After you add the label of a node pool here, this application pod can run only on nodes i
n the node pool.
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```

- Set nodeAffinity.
  You can also use nodeAffinity to schedule application pods. The following four scheduling policies are available:

  ```
  - requiredDuringSchedulingIgnoredDuringExecution
  ```

  If this policy is used, the pod must be deployed on a node that meets the requirements. If no nodes meet the requirements, the system retries until a node that meets the requirements is found. IgnoreDuringExecution indicates that if the tag of the node where the pod is deployed changes and does not meet the requirements, the pod keeps running on the node.

  ```
  - requiredDuringSchedulingRequiredDuringExecution
  ```

  If this policy is used, the pod must be deployed on a node that meets the requirements. If no nodes meet the requirements, the system retries until a node that meets the requirements is found. RequiredDuringExecution indicates that if the tag of the node where the pod is deployed changes and does not meet the requirements, the system selects another node that meets the requirements to deploy the pod.

  ```
  - preferredDuringSchedulingIgnoredDuringExecution
  ```

  If this policy is used, the pod is preferentially deployed on a node that meets the requirements. If no nodes meet the requirements, the system ignores these requirements.

  ```
  - preferredDuringSchedulingRequiredDuringExecution
  ```

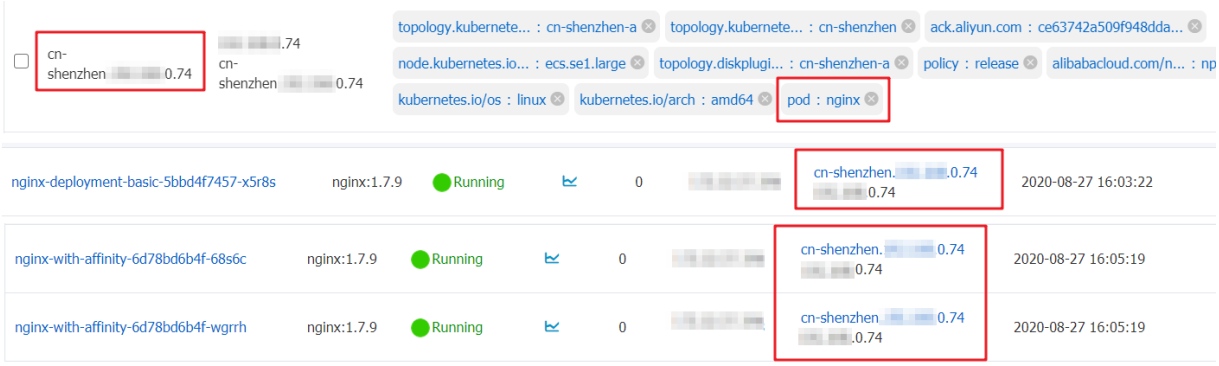
  If this policy is used, the pod is preferentially deployed on a node that meets the requirements. If no nodes meet the requirements, the system ignores these requirements. RequiredDuringExecution indicates that if the tag of a node changes and meets the requirements after the pod is deployed on another node, the system deploys the pod on the node that meets the requirements.

In the following example, the requiredDuringSchedulingIgnoredDuringExecution policy is used to ensure that the application runs on a node in the specified node pool.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-with-affinity
 labels:
  app: nginx-with-affinity
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx-with-affinity
 template:
  metadata:
   labels:
    app: nginx-with-affinity
  spec:
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
       - key: pod
         operator: In     # The application runs on a node that has the pod: nginx label.
         values:
         - nginx
   containers:
   - name: nginx-with-affinity
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```

Result

All application pods in the preceding examples are scheduled to the xxx.xxx.0.74 node, which is the node that has the pod: nignx label.



5.3.4. Configure custom kubelet parameters of a managed node pool功能发布后，文档再上线

Container Service for Kubernetes (ACK) allows you to configure custom kubelet parameters of a node pool. This way, you can configure custom kubelet parameters for the nodes in the node pool. For example, you can set kube-reserved and system-reserved to reserve computing resources and avoid exhaustion of node resources. When a large number of pods are deployed, you can modify kube-api-qps to improve the performance of kubelet to handle concurrent requests. This topic describes how to configure custom kubelet parameters of a node pool.

Prerequisites

- A professional managed Kubernetes cluster of Kubernetes 1.13 or later is created. Only professional Kubernetes clusters are supported. For more information, see Create a professional managed Kubernetes cluster.
- A managed node pool is created. For more information, see Create a managed node pool.

Context

kubelet is a core component of Kubernetes that runs as an agent on each node. kubelet runs processes by using binary files. kubelet is also the first service that starts in a Kubernetes cluster.

You can modify kubelet parameters based on your requirements. After the modifications are submitted, the nodes in a node pool will update kubelet in batches. Newly added nodes use the updated kubelet parameters.

Precautions

- Modifications to node pools and kubelet parameters are dependent on the custom object **NodePool** defined by Alibaba Cloud and all ConfigMaps in the kube-system namespace. To ensure the stability of node pools, we recommend that you do not modify these resources.
- Modifications to kubelet parameters cause kubelet processes to restart. During the restart, the node does not remain in the **ready** state and node events also change. If your service or monitoring is strongly dependent on the node state and events, modify kubelet parameters with caution.
- It requires about 1 minute to update kubelet parameters on a node. During the update, do not perform operations on the node. Otherwise, the update may fail and the kubelet parameters of the node may be rolled back to the previous settings. If an update or rollback fails, you can check the kubelet log or Submit a ticket to the Alibaba Cloud R&D team.
- Before you modify kubelet parameters, we recommend that you carefully read the related documentation and understand the parameters to avoid invalid settings. For a complete list of the kubelet parameters and their attributes, see kubelet.

Configure custom kubelet parameters of a managed node pool

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. On the **Node Pools** page, find the managed node pool that you want to manage and click **Configure kubelet** in the **Actions** column.
6. In the dialog box, set the parameters and click **OK**.

   The following types of parameters can be set:

   - qps and burst: Parameters of this type are used to configure traffic throttling of kubelet.
   - Resource and eviction configurations: Parameters of this type are used to configure and reserve resources, and evict nodes.

Type	Parameter	Value type	Example	Feature
qps and burst	event-qps	int	5	◦ If the value is greater than 0, it indicates the maximum number of events that can be generated per second. ◦ If you set this parameter to 0, it indicates that the number of events that can be generated per second is unlimited. Default value: 5.
	event-burst	int	10	The maximum number of events that can be reported. This allows event records to temporarily reach an upper limit without exceeding the number specified by event-qps. This parameter is used only if the value of `--event-qps` is greater than 0. Default value: 10.
	kube-api-qps	int	5	The maximum number of queries per second (QPS) that can be sent to the Kubernetes API server. Default value: 5.
	kube-api-burst	int	10	The maximum number of requests that can be forwarded to the Kubernetes API server per second. Default value: 10.
	registry-qps	int	5	◦ If the value of `--registry-qps` is greater than 0, it indicates the QPS upper limit of image pulls. ◦ If you set this parameter to 0, it indicates that the QPS of image pulls is unlimited. Default value: 5.
	registry-burst	int	10	The maximum number of bursty image pulls. This allows image pulls to temporarily reach the specified maximum number without exceeding the number specified by registry-qps. This parameter is used only if the value of `--registry-qps` is greater than 0. Default value: 10.
	serialize-image-pulls	bool	true	Specifies whether only one image can be pulled at a time. Default value: true.

Type	Parameter	Value type	Example	Feature
Resource and eviction configurations	eviction-hard	map[String]String	memory.available: 300Minodefs.available: 10%nodefs.inodesFree: 5%imagefs.available: 15%imagefs.inodesFree: 5%pid.available: 100	A set of hard eviction thresholds for pods. Default value of ACK: `--eviction-hard=imagefs.available<15%,memory.available<300Mi,nodefs.available<10%,nodefs.inodesFree<5%` .
	eviction-soft	map[String]String	Refer to the sample value of eviction-hard	A set of soft eviction thresholds for pods. ACK does not provide default settings.
	kube-reserved	map[String]String	cpu:100mmemory: 300Miephemeral-storage: 1Gipid: 10000	A set of configurations that specify reserved resources for the Kubernetes system components. Default value of ACK: `--system-reserved=memory=300Mi-system-reserved=pid=10000` .
	system-reserved	map[String]String	Refer to the sample value of kube-reserved	A set of configurations that specify reserved resources for the system. Default value of ACK: `--kube-reserved=memory=400Mi--kube-reserved=pid=10000` .

After kubelet is updated for all nodes in a node pool, the node pool changes to the **Active** state. You can run the **describe node** command to verify that the modifications have been applied to resource reservation parameters. To verify that the modifications have been applied to qps and burst parameters, you must perform stress testing or run commands to check the kubelet log. For example, you can run the `journalctl -u kubelet` command to check the kubelet log.

Update policy of kubelet parameters

10% of the nodes in a node pool are updated at a time until all the nodes are updated. If the update fails on a node, the update is terminated and this node is rolled back to the previous kubelet configuration.

For example, a node pool has five nodes. In this case, one node is updated at a time. Five updates are required to update all the nodes in the node pool. If the update fails on a node, the update is terminated and this node is rolled back to the previous kubelet configuration. During the rollback, this node remains in the **Updating** state. If you want to continue updating this node, we recommend that you first locate and fix the issue. If a node pool has 100 nodes, 10 nodes are updated at a time. 10 updates are required to update all the nodes based on the preceding description.

5.4. FAQ about node management

This topic provides answers to some frequently asked questions about node management.

- How do I manually upgrade the kernel of the GPU nodes in a cluster?
- How do I resolve the issue that no container is started on a GPU node?
- FAQ about adding nodes to a cluster
- The "drain-node job execute timeout" error appears during node removal

How do I manually upgrade the kernel of the GPU nodes in a cluster?

You can manually upgrade the kernel of the GPU nodes in a cluster by performing the following steps:

> **Note** The current kernel version is earlier than `3.10.0-957.21.3`.
> Confirm the kernel version to which you want to upgrade. Proceed with caution.
> You can perform the following steps to upgrade the kernel and NVIDIA driver.

1. Connect to Kubernetes clusters by using kubectl.
2. Set the GPU node that you want to manage to the unschedulable state. In this example, the node cn-beijing.i-2ze19qyi8votgjz12345 is used.

   ```
   kubectl cordon cn-beijing.i-2ze19qyi8votgjz12345
   node/cn-beijing.i-2ze19qyi8votgjz12345 already cordoned
   ```

3. Migrate the pods on the GPU node to other nodes.

   ```
   kubectl drain cn-beijing.i-2ze19qyi8votgjz12345 --grace-period=120 --ignore-daemonsets=true
   node/cn-beijing.i-2ze19qyi8votgjz12345 cordoned
   WARNING: Ignoring DaemonSet-managed pods: flexvolume-9scb4, kube-flannel-ds-r2qmh, kube-proxy-worker-l62sf, logtail-ds-f9vbg
   pod/nginx-ingress-controller-78d847fb96-5fkkw evicted
   ```

4. Uninstall the existing nvidia-driver.

   > **Note** In this example, the driver of version 384.111 is uninstalled. If your driver version is not 384.111, download the installation package of your driver from the official NVIDIA website and replace version `384.111` with your actual driver version number.

   i. Log on to the GPU node and run the `nvidia-smi` command to check the driver version.

   ```
   nvidia-smi -a | grep 'Driver Version'
   Driver Version                  : 384.111
   ```

   ii. Download the driver installation package.

   ```
   cd /tmp/
   curl -O https://cn.download.nvidia.cn/tesla/384.111/NVIDIA-Linux-x86_64-384.111.run
   ```

   > **Note** The installation package is required to uninstall the driver.

   iii. Uninstall the existing driver.

   ```
   chmod u+x NVIDIA-Linux-x86_64-384.111.run
   ./NVIDIA-Linux-x86_64-384.111.run --uninstall -a -s -q
   ```

5. Upgrade the kernel.

   Upgrade the kernel based on your requirements.

6. Restart the GPU node.

```
reboot
```

7. Log on to the GPU node to install the corresponding kernel-devel package.

```
yum install -y kernel-devel-$(uname -r)
```

8. Go to the official NVIDIA website to download the required driver and install it on the GPU node. In this example, the driver of version 410.79 is used.

```
cd /tmp/
curl -O https://cn.download.nvidia.cn/tesla/410.79/NVIDIA-Linux-x86_64-410.79.run
chmod u+x NVIDIA-Linux-x86_64-410.79.run
sh ./NVIDIA-Linux-x86_64-410.79.run -a -s -q
warm up GPU
nvidia-smi -pm 1 || true
nvidia-smi -acp 0 || true
nvidia-smi --auto-boost-default=0 || true
nvidia-smi --auto-boost-permission=0 || true
nvidia-modprobe -u -c=0 -m || true
```

9. Check the */etc/rc.d/rc.local* file to verify that the following configurations are included. Otherwise, add the following configurations to the file.

```
nvidia-smi -pm 1 || true
nvidia-smi -acp 0 || true
nvidia-smi --auto-boost-default=0 || true
nvidia-smi --auto-boost-permission=0 || true
nvidia-modprobe -u -c=0 -m || true
```

10. Restart kubelet and Docker.

```
service kubelet stop
service docker restart
service kubelet start
```

11. Set the GPU node to schedulable.

```
kubectl uncordon cn-beijing.i-2ze19qyi8votgjz12345
node/cn-beijing.i-2ze19qyi8votgjz12345 already uncordoned
```

12. Run the following command in the nvidia-device-plugin container to check the version of the driver installed on the GPU node.

```
kubectl exec -n kube-system -t nvidia-device-plugin-cn-beijing.i-2ze19qyi8votgjz12345 nvidia-smi
Thu Jan 17 00:33:27 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.79     Driver Version: 410.79     CUDA Version: N/A     |
|-------------------------------+----------------------+----------------------+
| GPU  Name      Persistence-M| Bus-Id      Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|  0  Tesla P100-PCIE...  On   | 00000000:00:09.0 Off |                    0 |
| N/A  27C   P0   28W / 250W |    0MiB / 16280MiB |    0%     Default |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
| GPU     PID   Type   Process name                             Usage      |
|=============================================================================|
| No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Note If you run the `docker ps` command and find that no container is started on the GPU node, see Failed to start a container on the GPU node.

How do I resolve the issue that no container is started on a GPU node?

When you restart kubelet and Docker on GPU nodes where specific Kubernetes versions are installed, you may find that no container is started on the nodes after the restart.

```
service kubelet stop
Redirecting to /bin/systemctl stop kubelet.service
service docker stop
Redirecting to /bin/systemctl stop docker.service
service docker start
Redirecting to /bin/systemctl start docker.service
service kubelet start
Redirecting to /bin/systemctl start kubelet.service
docker ps
CONTAINER ID     IMAGE         COMMAND         CREATED         STATUS         PORTS         NAMES
```

Run the following command to check the cgroup driver.

```
docker info | grep -i cgroup
Cgroup Driver: cgroupfs
```

The output shows that the cgroup driver is set to cgroupfs.
To resolve the issue, perform the following steps:

1. Create a copy of */etc/docker/daemon.json*. Then, run the following commands to update */etc/docker/daemon.json*.

```
cat >/etc/docker/daemon.json <<-EOF
{
  "default-runtime": "nvidia",
  "runtimes": {
    "nvidia": {
      "path": "/usr/bin/nvidia-container-runtime",
      "runtimeArgs": []
    }
  },
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "10"
  },
  "oom-score-adjust": -1000,
  "storage-driver": "overlay2",
  "storage-opts":["overlay2.override_kernel_check=true"],
  "live-restore": true
}
EOF
```

2. Run the following commands in sequence to restart Docker and kubelet.

```
service kubelet stop
Redirecting to /bin/systemctl stop kubelet.service
service docker restart
Redirecting to /bin/systemctl restart docker.service
service kubelet start
Redirecting to /bin/systemctl start kubelet.service
```

3. Run the following command to verify that the cgroup driver is set to systemd.

```
docker info | grep -i cgroup
Cgroup Driver: systemd
```

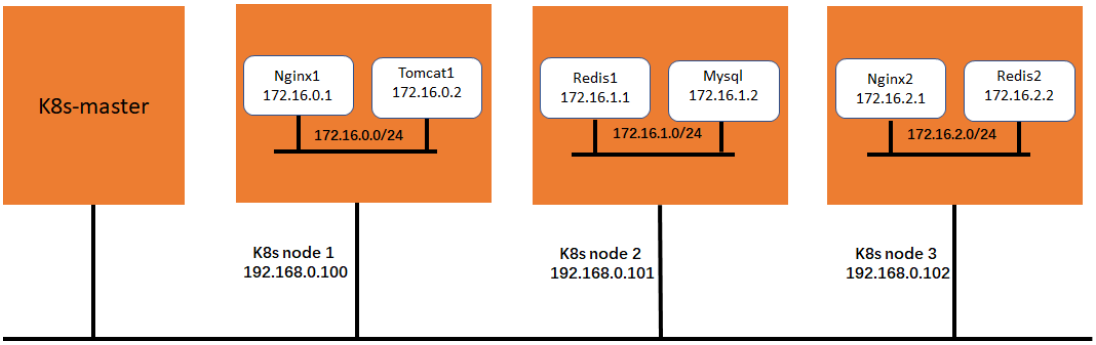
6.Network

6.1. Overview

Container Service for Kubernetes (ACK) provides stable and high-performance container networks by integrating Kubernetes networking, Virtual Private Cloud (VPC), and Server Load Balancer (SLB). This topic describes the important terms involved in ACK cluster networking and Alibaba Cloud network infrastructure, such as container network interface (CNI), Service, Ingress, and DNS Service Discovery (DNS-SD). Understanding these terms helps you optimize application deployment models and network access methods.

CNI

Container network model



Containerized applications deploy multiple workloads on a node. Each workload requires a unique network namespace. To avoid network conflicts, each pod must have a unique network namespace. To enable the application that runs on a pod to communicate with other networks, the pod must be able to access other networks. Container networking has the following features:

- A pod has a unique network namespace and IP address. Applications that run on different pods can listen on the same port without conflicts.
- Pods can access each other by using their IP addresses.
  In a cluster, a pod can communicate with other applications by using a unique IP address.
  - Pods can access each other within a cluster.
  - Pods can access Elastic Compute Service (ECS) instances that are deployed in the same VPC.
  - ECS instances can access pods that are deployed in the same VPC.

> **Note** To ensure that pods can access ECS instances in the same VPC and ECS instances can access pods in the same VPC, you must configure correct security group rules. For more information about how to configure security group rules, see Add security group rules.

ACK provides two network plug-ins to help you implement container networking: Flannel and Terway. The two network plug-ins adopt different network models, which have the following features:

For more information about the CIDR blocks of VPC and Kubernetes clusters, see 专有网络VPC网段和Kubernetes网段关系.

For more information about how to select a network plug-in based on your business requirements, see Use the Terway plug-in.

Terway | Flannel

Service

Cloud-native applications usually require agile iterations and rapid elasticity. Single containers and the related network resources have short lifecycles. This necessitates stable addresses and automatic load balancing to achieve rapid elasticity. In ACK, a Service provides a set of pods with a stable address and balances the loads across the pods. How a Service works

- When you create a Service, ACK assigns a stable IP address to the Service.
- You use the selector field to select a set of pods for the Service. The Service serves as a load balancer and forwards traffic that is sent to the Service IP and port to the IP addresses and ports of the selected pods.

ACK provides the following types of Services to handle requests from different sources and clients:

- *ClusterIP*
  A ClusterIP Service is used for access within the cluster. If you want your application to provide services within the cluster, create a ClusterIP Service.

  > **Note** The ClusterIP type is the default Service type.

- *NodePort*
  A NodePort Service is used to expose an application to the Internet. You can use the IP address and port of a cluster node to expose your application. This way, your application can be accessed by using the node IP address and port.

- *LoadBalancer*
  A LoadBalancer Service is also used to expose an application to the Internet. However, an SLB instance is used to expose your application, which provides higher availability and performance than a NodePort Service. For more information about how to use a LoadBalancer Service to expose an application, see Use an existing SLB instance to expose an application and Use an automatically created SLB instance to expose an application.

- *Headless Service*
  A Headless Service is defined by setting the clusterIP field to None in the Service configuration file. A Headless Service has no stable virtual IP address (VIP). When a client accesses the domain of the Service, DNS returns the IP addresses of all backend pods. The client must use DNS load balancing to balance the loads across pods.

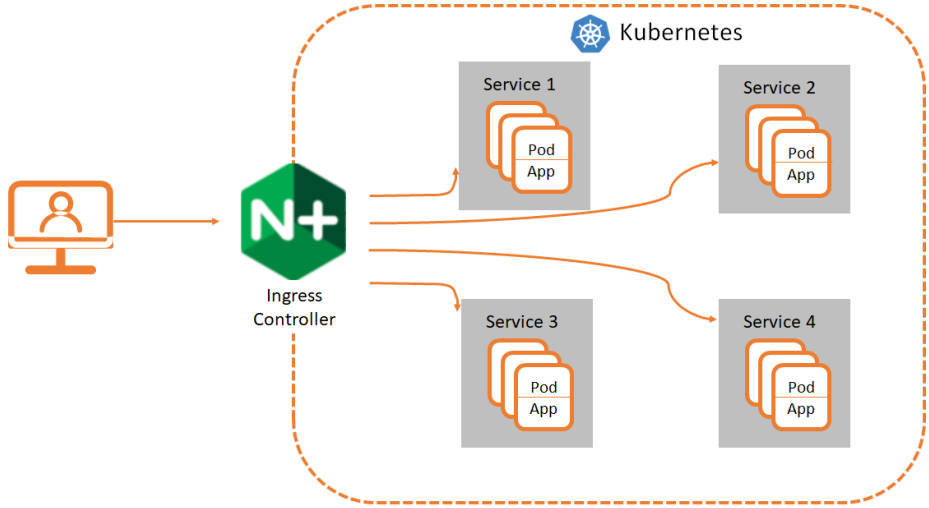
- *ExternalName*
  An ExternalName Service is used to map an external domain to a Service within the cluster. For example, you can map the domain of an external database to a Service name within the cluster. This allows you to access the database within the cluster by using the Service name.

For more information about how to configure a LoadBalancer Service, see Considerations for configuring a LoadBalancer type Service.

Ingress

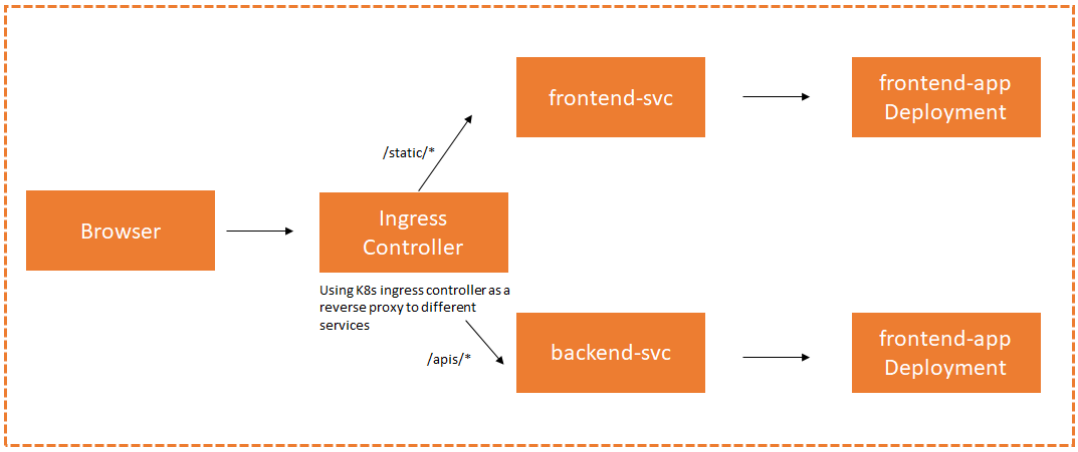
In ACK clusters, Services support Layer 4 load balancing. However, Ingresses manage external access to Services in the cluster at Layer 7. You can use Ingresses to configure different Layer 7 forwarding rules. For example, you can forward requests to different Services based on domain name or access path. This way, Layer 7 load balancing is implemented. For more information, see Ingress overview.

The relationship between Ingress and Service



Example

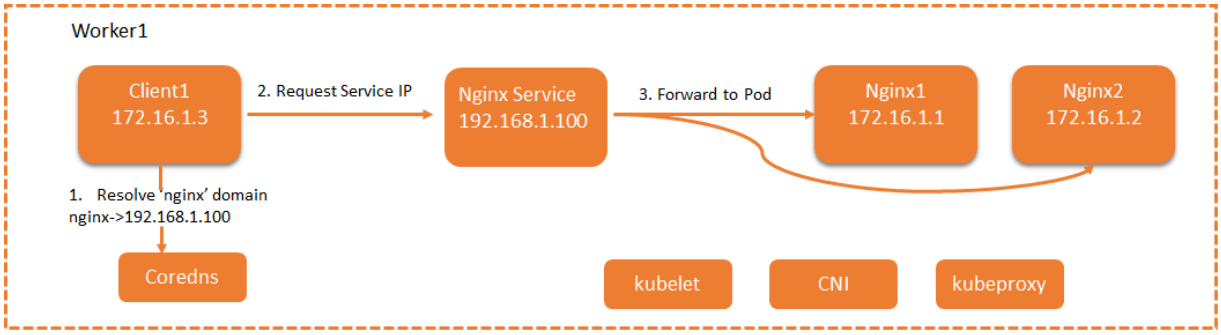
In the common architecture pattern that separates frontend and backend, different access paths are used to access frontend and backend services. In this scenario, Ingresses can be used to implement Layer 7 load balancing across different application services.



DNS Service Discovery

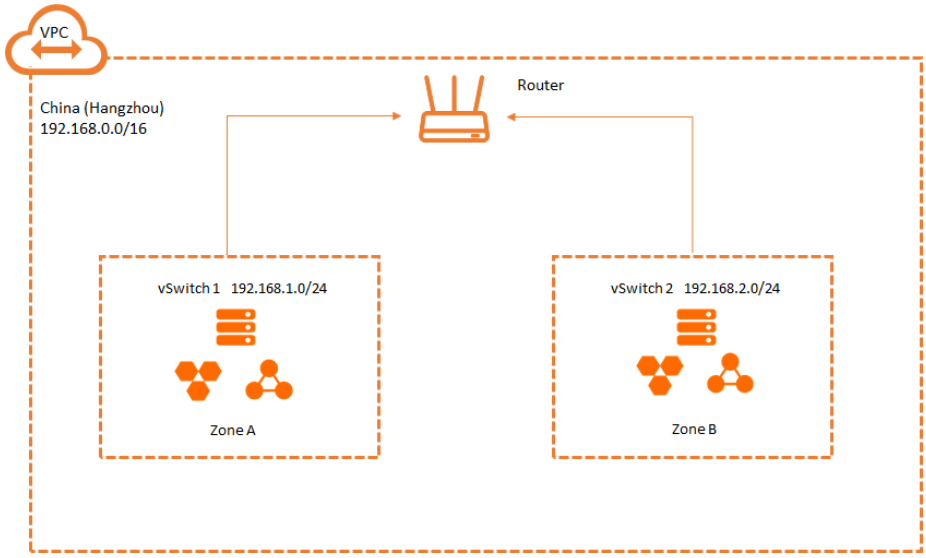
ACK uses DNS for service discovery. For example, a client can resolve the DNS name of a Service to obtain the cluster IP of the Service, or resolve the DNS name of a pod that is created by a StatefulSet to obtain the IP address of the pod. Using DNS for service discovery allows you to invoke applications by using DNS names, regardless of IP addresses or deployment environments.

CoreDNS automatically converts the DNS name of a Service to the IP address of the Service. This allows you to use the same connection address in different deployment environments. For more information about how to configure and optimize DNS resolution in a cluster, see Introduction and configuration of the DNS service in ACK clusters.



Infrastructure

- *VPC*
  VPC is a type of private network provided by Alibaba Cloud. VPCs are logically isolated from one other. You can create and manage cloud services in VPCs, such as ECS instances, ApsaraDB RDS, and SLB instances.
  Each VPC consists of one vRouter, at least one private CIDR block, and at least one vSwitch.



  - Private CIDR blocks
    When you create a VPC and a vSwitch, you must specify the private IP address range for the VPC in CIDR notation.
    You can use the standard private CIDR blocks listed in the following table and their subsets as CIDR blocks for your VPCs. For more information, see Plan and design a VPC. For more information, see the Plan and design a VPC topic in *User Guide*.

CIDR blocks	Number of available private IP addresses (system reserved ones excluded)
192.168.0.0/16	65,532
172.16.0.0/12	1,048,572
10.0.0.0/8	16,777,212

- vRouters
  A vRouter is the hub of a VPC and serves as a gateway between the VPC and other networks. After a VPC is created, a vRouter is automatically created for the VPC. Each vRouter is associated with a route table.
  For more information, see 路由表概述.
  For more information, see the Route table overview topic in *User Guide*.
- vSwitches
  A vSwitch is a basic network component that connects different cloud resources in a VPC. After you create a VPC, you can create a vSwitch to divide your VPC into multiple subnets. vSwitches deployed in a VPC can communicate with each other over the private network. You can deploy your applications in vSwitches that belong to different zones to improve service availability.
  For more information, see vSwitches.
  For more information, see the Create a vSwitch topic in *User Guide*.

- *SLB*
  After you connect ECS instances to an SLB instance, SLB uses virtual IP addresses (VIPs) to virtualize the ECS instances and adds the ECS instances to an application service pool. The application service pool features high performance and high availability. Client requests are distributed across the ECS instances based on forwarding rules.
  SLB checks the health status of the ECS instances and automatically removes unhealthy ECS instances from the pool to eliminate single points of failure. This improves the availability of your applications. You can also use SLB to defend your applications against distributed denial of service (DDoS) attacks.
  SLB consists of the following components:
  - SLB instance
    An SLB instance is a running SLB service entity that receives and distributes traffic to backend servers. To use SLB, you must create an SLB instance and create at least one listener and connect at least two ECS instances to the SLB instance.
  - Listener
    A listener checks client requests, forwards the requests to backend servers, and performs health checks on backend servers.
  - Backend server
    Backend servers are ECS instances that receive client requests that are distributed from the frontend SLB service. You can add ECS instances to a server pool, or use VServer groups or primary/secondary server groups to add and manage ECS instances in batches.

Related information

- Manage service quotas
- 使用网络策略Network Policy
- Plan CIDR blocks for an ACK cluster
- Use the Terway plug-in
- Create a security group

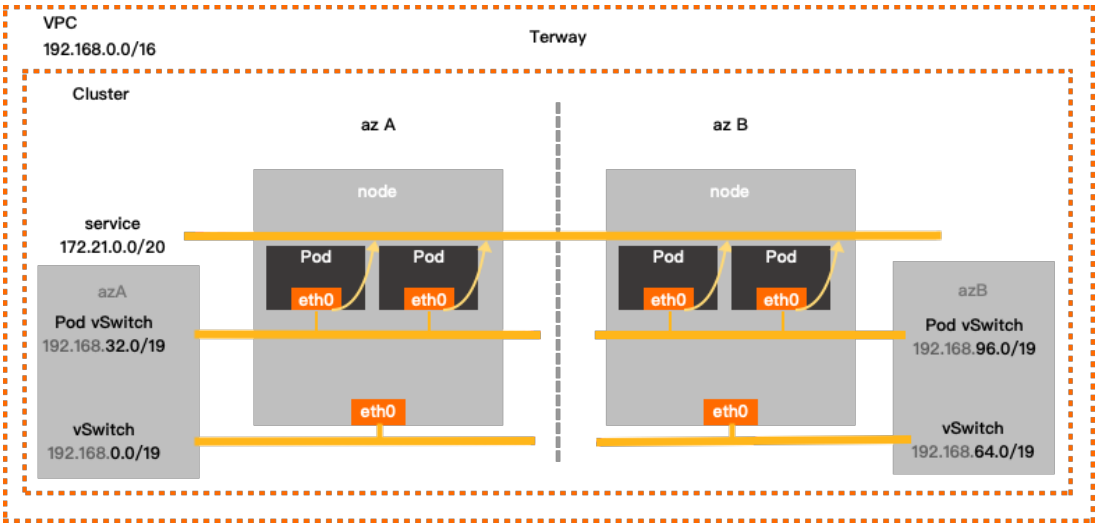
6.2. Plan CIDR blocks for an ACK cluster

When you create a Container Service for Kubernetes (ACK) cluster, you must specify a virtual private cloud (VPC), vSwitches, the CIDR block of pods, and the CIDR block of Services. Therefore, we recommend that you plan the IP address of each Elastic Compute Service (ECS) instance in the cluster, the CIDR block of pods, and the CIDR block of Services before you create an ACK cluster. This topic describes how to plan CIDR blocks for an ACK cluster deployed in a VPC and how each CIDR block is used.

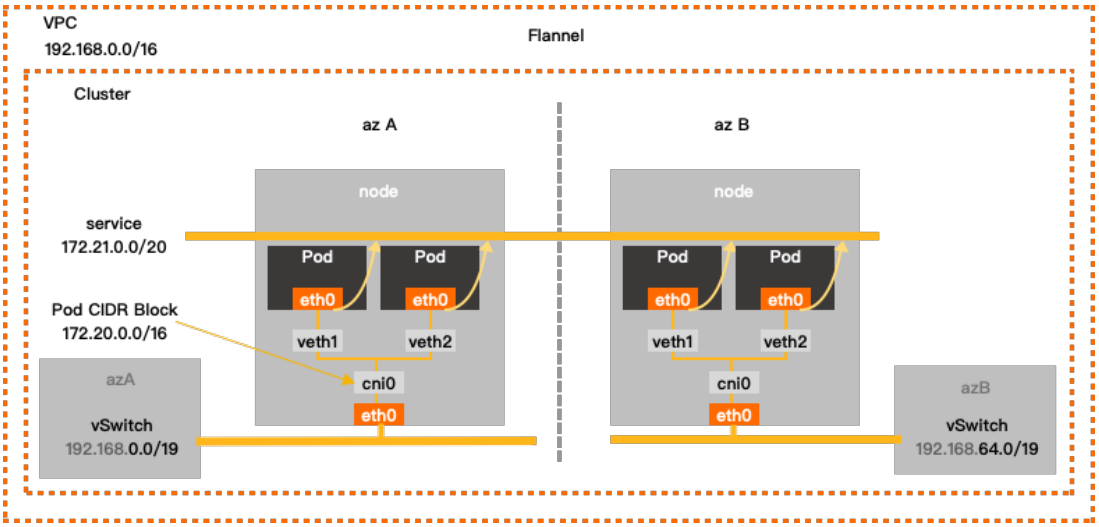
VPC-related CIDR blocks and cluster-related CIDR blocks

Before you create a VPC, you must plan the CIDR block of the VPC and the CIDR blocks of vSwitches in the VPC. Before you create an ACK cluster, you must plan the CIDR block of pods and the CIDR block of Services. ACK supports the Terway and Flannel plug-ins. The following figure shows the network architecture of an ACK cluster that uses Terway or Flannel.

Terway



Flannel



To install Terway or Flannel in an ACK cluster, you must specify CIDR blocks and other parameters as described in the following table.

Parameter	Terway	Flannel
VPC	When you create a VPC, you must select a CIDR block for the VPC. Valid values: 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.	

Parameter	Terway	Flannel
VSwitch	The IP addresses of ECS instances are assigned from the vSwitch. This way, the nodes in a cluster can communicate with each other. The CIDR blocks that you specify when you create vSwitches in the VPC must be subsets of the VPC CIDR block. This means that the CIDR blocks of vSwitches must be smaller than or the same as the VPC CIDR block. When you set this parameter, take note of the following items: • Select one or more vSwitches in the VPC. • IP addresses from the CIDR block of a vSwitch are allocated to the ECS instances that are attached to the vSwitch. • You can create multiple vSwitches in a VPC. However, the CIDR blocks of these vSwitches cannot overlap with each other. • The vSwitch and the pod vSwitches must be in the same zone. For more information about zones, see Regions and zones.	The IP addresses of ECS instances are assigned from the vSwitch. This way, the nodes in a cluster can communicate with each other. The CIDR blocks that you specify when you create vSwitches in the VPC must be subsets of the VPC CIDR block. This means that the CIDR blocks of vSwitches must be smaller than or the same as the VPC CIDR block. When you set this parameter, take note of the following items: • Select one or more vSwitches in the VPC. • IP addresses from the CIDR block of a vSwitch are allocated to the ECS instances that are attached to the vSwitch. • You can create multiple vSwitches in a VPC. However, the CIDR blocks of these vSwitches cannot overlap with each other.
Pod VSwitch	The IP addresses of pods are assigned from the CIDR block of the pod vSwitches. This way, the pods can communicate with each other. A pod is a group of containers in a Kubernetes cluster. Each pod has an IP address. The CIDR blocks that you specify when you create pod vSwitches in the VPC must be subsets of the VPC CIDR block. When you set this parameter, take note of the following items: • Select one or more vSwitches in the VPC. • In an ACK cluster that has Terway installed, the IP addresses of pods are allocated from pod vSwitches. • The CIDR blocks of pod vSwitches cannot overlap with the CIDR blocks of **vSwitches**. • The CIDR blocks of pod vSwitches cannot overlap with the CIDR block specified by **Service CIDR**. • The vSwitch and the pod vSwitches must be in the same zone. For more information about zones, see Regions and zones.	You do not need to set this parameter if you install Flannel in an ACK cluster.

Parameter	Terway	Flannel
Pod CIDR Block	You do not need to set this parameter if you install Terway in an ACK cluster.	The IP addresses of pods are allocated from the pod CIDR block. This way, the pods can communicate with each other. A pod is a group of containers in a Kubernetes cluster. Each pod has an IP address. When you set this parameter, take note of the following items: • Enter a CIDR block in the Pod CIDR Block field. • The CIDR block of pods cannot overlap with the CIDR blocks of **vSwitches**. • The CIDR block of pods cannot overlap with the CIDR block specified by **Service CIDR**. For example, if the VPC CIDR block is 172.16.0.0/12, the CIDR block of pods cannot be 172.16.0.0/16 or 172.17.0.0/16, because these CIDR blocks are subsets of 172.16.0.0/12.
Service CIDR	The CIDR block of Services. Service is an abstraction in Kubernetes. Each `ClusterIP` Service has an IP address. When you set this parameter, take note of the following items: • The IP address of a Service is effective only within the Kubernetes cluster. • The CIDR block of Services cannot overlap with the CIDR blocks of **vSwitches**. • The CIDR block of Services cannot overlap with the CIDR blocks of **Pod vSwitches**.	The CIDR block of Services. Service is an abstractino in Kubernetes. Each `ClusterIP` Service has an IP address. When you set this parameter, take note of the following items: • The IP address of a Service is effective only within the Kubernetes cluster. • The CIDR block of Services cannot overlap with the CIDR blocks of **vSwitches**. • The CIDR block of Services cannot overlap with **Pod CIDR Block**.

Considerations

Network planning

To use Kubernetes clusters that are supported by ACK on Alibaba Cloud, you must first set up networks for the clusters based on the cluster size and business scenarios. You can refer to the following tables to set up networks for a cluster. Change specifications as needed in unspecified scenarios.

Cluster size	Scenario	VPC	Zone
≤ 100 nodes	Regular business	Single VPC	1
Unlimited	Cross-zone deployment is required	Single VPC	≥ 2
Unlimited	High reliability and cross-region deployment are required	Multiple VPCs	≥ 2

The following tables describe how to plan CIDR blocks for clusters that use Flannel or Terway:

- **A cluster that uses Flannel**

VPC CIDR block	vSwitch CIDR block	Pod CIDR block	Service CIDR block	Maximum number of pod IP addresses
192.168.0.0/16	192.168.0.0/24	172.20.0.0/16	172.21.0.0/20	65536

- **A cluster that uses Terway**
  - One elastic network interface (ENI) to Each Pod mode or IPVLAN mode is enabled

VPC CIDR block	vSwitch CIDR block	CIDR block of pod vSwitches	Service CIDR block	Maximum number of pod IP addresses
192.168.0.0/16	192.168.0.0/19	192.168.32.0/19	172.21.0.0/20	8192

  - **Multi-zone deployment**

VPC CIDR block	vSwitch CIDR block	CIDR block of pod vSwitches	Service CIDR block	Maximum number of pod IP addresses
192.168.0.0/16	Zone I 192.168.0.0/19	192.168.32.0/19	172.21.0.0/20	8192
	Zone J 192.168.64.0/19	192.168.96.0/19		8192

Plan the network of a VPC

Plan CIDR blocks for clusters

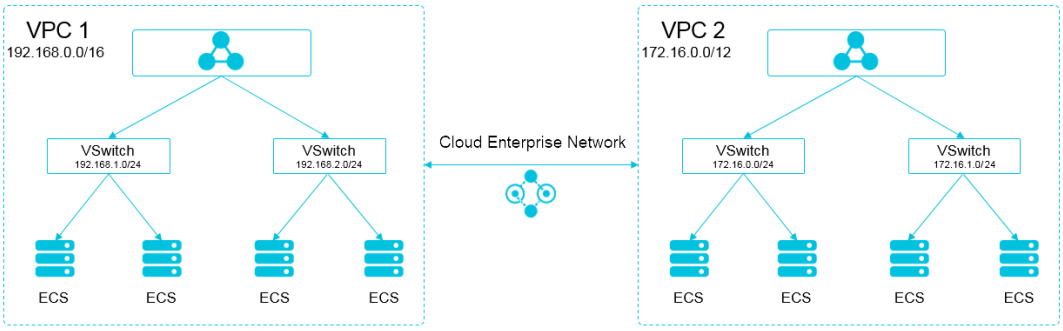
How to plan CIDR blocks

- **Scenario 1: one VPC and one Kubernetes cluster**
  This is the simplest scenario. The CIDR block of a VPC is specified when you create the VPC. When you create a cluster in the VPC, make sure that the CIDR block of pods and the CIDR block of Services do not overlap with the VPC CIDR block.
- **Scenario 2: one VPC and multiple Kubernetes clusters**
  You want to create more than one cluster in a VPC.
  - The CIDR block of the VPC is specified when you create the VPC. When you create clusters in the VPC, make sure that the VPC CIDR block, Service CIDR block, and pod CIDR block of each cluster do not overlap with each other.
  - The Service CIDR blocks of the clusters can overlap with each other. However, the pod CIDR blocks cannot overlap with each other.
  - In the default network mode (Flannel), the packets of pods must be forwarded by the VPC router. ACK automatically generates a route table for each destination pod CIDR block on the VPC router.

  > **Note** In this case, a pod in one cluster can communicate with the pods and ECS instances in another cluster. However, the pod cannot communicate with the Services in another cluster.

- **Scenario 3: two connected VPCs**

If two VPCs are connected, you can use the route table of one VPC to specify the packets that you want to send to the other VPC. The CIDR block of VPC 1 is 192.168.0.0/16 and the CIDR block of VPC 2 is 172.16.0.0/12, as shown in the following figure. You can use the route table of VPC 1 to forward all packets that are destined for 172.16.0.0/12 to VPC 2.



Connected VPCs

VPC	CIDR block	Destination CIDR block	Destination VPC
VPC 1	192.168.0.0/16	172.16.0.0/12	VPC 2
VPC 2	172.16.0.0/12	192.168.0.0/16	VPC 1

In this scenario, make sure that the following conditions are met when you create a cluster in VPC 1 or VPC 2:

- The CIDR blocks of the cluster do not overlap with the CIDR block of VPC 1.
- The CIDR blocks of the cluster do not overlap with the CIDR block of VPC 2.
- The CIDR blocks of the cluster do not overlap with those of other clusters.
- The CIDR blocks of the cluster do not overlap with those of pods.
- The CIDR blocks of the cluster do not overlap with those of Services.

In this example, you can set the pod CIDR block of the cluster to a subset of 10.0.0.0/8.

> **Note** All IP addresses in the destination CIDR block of VPC 2 can be considered in use. Therefore, the CIDR blocks of the cluster cannot overlap with the destination CIDR block.

To access pods in VPC 1 from VPC 2, you must configure a route in VPC 2. The route must point to the pod CIDR block of a cluster in VPC 1.

- **Scenario 4: a VPC connected to a data center**
  If a VPC is connected to a data center, packets of specific CIDR blocks are routed to the data center. In this case, the pod CIDR block of a cluster in the VPC cannot overlap with these CIDR blocks. To access pods in the VPC from the data center, you must configure a route in the data center to enable VBR-to-VPC peering connection.

Related information

- Overview
- Use the Terway plug-in
- 使用网络策略Network Policy
- FAQ about network management

6.3. Container network

6.3.1. Use the Terway plug-in

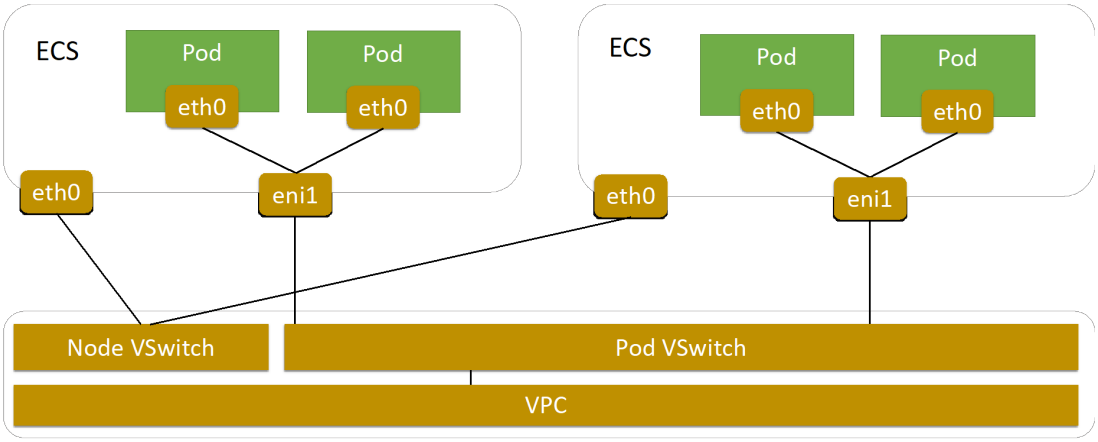
Terway is an open source Container Network Interface (CNI) plug-in developed by Alibaba Cloud. Terway is built on Virtual Private Cloud (VPC) and allows you to regulate how containers communicate with each other by using standard Kubernetes network policies. You can use Terway to enable intercommunication within a Kubernetes cluster. This topic describes how to use Terway in a Container Service for Kubernetes (ACK) cluster.

Context

Terway is a network plug-in developed by Alibaba Cloud for ACK. Terway allows you to configure networks for pods by associating Alibaba Cloud elastic network interfaces (ENIs) with the pods. Terway allows you to use standard Kubernetes network policies to regulate how containers communicate with each other. In addition, Terway is compatible with Calico network polices.

In a cluster that has Terway installed, each pod has a separate network stack and is assigned a separate IP address. Pods on the same Elastic Compute Service (ECS) instance communicate with each other by forwarding packets within the ECS instance. Pods on different ECS instances communicate with each other through an ENI in the VPC where the ECS instances are deployed. Tunneling technologies such as Virtual Extensible Local Area Network (VXLAN) are not required to encapsulate packets. This improves communication efficiency.

How Terway works



Terway and Flannel

When you create an ACK cluster, you can choose one of the following network plug-ins:

- Terway: a network plug-in developed by Alibaba Cloud for ACK. Terway allows you to assign ENIs to containers and use standard Kubernetes network policies to regulate how containers communicate with each other. Terway also supports bandwidth throttling on individual containers. Terway uses flexible IP Address Management (IPAM) policies to allocate IP addresses to containers. This prevents the waste of IP addresses. If you do not want to use network policies, you can select Flannel as the network plug-in. Otherwise, we recommend that you select Terway.
- Flannel: an open source CNI plug-in, which is simple and stable. You can use Flannel together with Alibaba Cloud VPC. This way, your clusters and containers can run in high-performance and stable networks. However, Flannel provides only basic features. It does not support standard Kubernetes network policies. For more information, see Flannel.

Item	Terway	Flannel
Performance	The IP address of each pod in an ACK cluster is assigned from the CIDR block of the VPC where the cluster is deployed. Therefore, you do not need to use the Network Address Translation (NAT) service to translate IP addresses. This prevents the waste of IP addresses. In addition, each pod in the cluster can use an exclusive ENI.	Flannel works together with Alibaba Cloud VPC. The CIDR block of pods that you specify must be different from that of the VPC where the cluster is deployed. Therefore, the NAT service is required and some IP addresses may be wasted.
Security	Terway supports network policies.	Flannel does not support network policies.
IP address management	Terway allows you to assign IP addresses on demand. You do not have to assign CIDR blocks by node. This prevents the waste of IP addresses.	You must assign CIDR blocks by node. In large-scale clusters, a significant number of IP addresses may be wasted.
SLB	Server Load Balancer (SLB) directly forwards network traffic to pods. You can upgrade the pods without service interruptions.	SLB forwards network traffic to the NodePort Service. Then, the NodePort Service routes the network traffic to pods.

Precaution

- To use the Terway plug-in, we recommend that you use ECS instances of higher specifications and later types, such as ECS instances that belong to the g5 or g6 instance families with at least 8 CPU cores. For more information, see Instance families.
- The maximum number of pods that each node supports is based on the number of ENIs assigned to the node.
  - Maximum number of pods supported by each shared ENI = (Number of ENIs supported by each ECS instance - 1) × Number of private IP addresses supported by each ENI
  - Maximum number of pods supported by each exclusive ENI = Number of ENIs supported by each ECS instance -1

> **Note** You can view the number of ENIs supported by each ECS instance in the **Instance Type** section when you create or expand a cluster. For more information about how to create a cluster, see Create a professional managed Kubernetes cluster.

Step 1: Plan the cluster network

When you create an ACK cluster, you must specify a VPC, vSwitches, the CIDR block of pods, and the CIDR block of Services. If you want to install the Terway plug-in, you must first create a VPC and two vSwitches in the VPC. The two vSwitches must be in the same zone. For more information about how to plan the network for a cluster that uses Terway, see Plan CIDR blocks for an ACK cluster.

You can refer to the following table to assign CIDR blocks for a cluster that uses Terway.

VPC CIDR block	vSwitch CIDR block	CIDR block of pod vSwitch	Service CIDR
192.168.0.0/16	192.168.0.0/19	192.168.32.0/19	172.21.0.0/20

Note

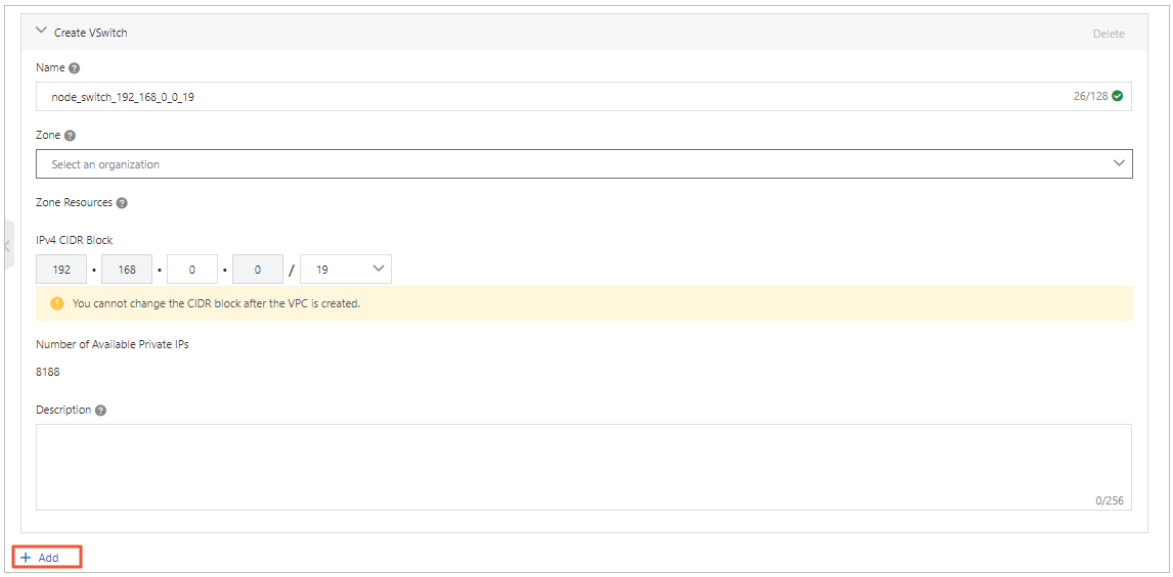
- The IP addresses within the CIDR block of the vSwitch are assigned to nodes.
- The IP addresses within the CIDR block of the pod vSwitch are assigned to pods.

The following example describes how to create a VPC and two vSwitches. The CIDR blocks in the preceding table are assigned in this example.

1. Log on to the VPC console.
2. In the top navigation bar, select the region where you want to deploy the VPC and click **Create VPC**.

   Note The VPC must be deployed in the same region as the cloud resources that you want to deploy.

3. On the **Create VPC** page, set **Name** to *vpc_192_168_0_0_16* and enter *192.168.0.0/16* in the **IPv4 CIDR Block** field.
4. In the **Create VSwitch** section, set the name to *node_switch_192_168_0_0_19*, select a zone for the vSwitch, and then set **IPv4 CIDR Block** to **192.168.0.0/19**. Click **Add** to create the pod vSwitch.

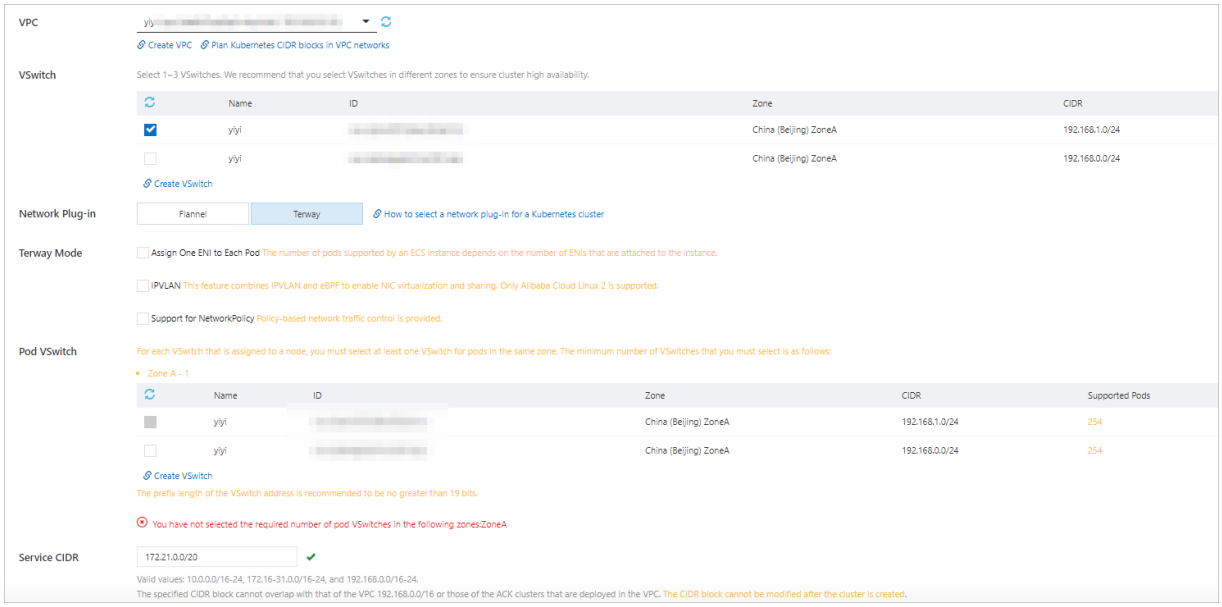


Notice Make sure that the two vSwitches are in the same zone.

5. In the **Create VSwitch** section, configure the pod vSwitch. Set the name to **pod_switch_192_168_32_0_19** and IPv4 CIDR Block to **192.168.32.0/19**.
6. Click **OK**.

Step 2: Configure networks for a cluster that uses Terway

To install Terway in a cluster and configure networks for the cluster, set the following parameters.



Note A standard managed Kubernetes cluster is used as an example to show how to configure networks for a cluster that uses Terway as the network plug-in. For more information about how to create a cluster, see 创建Kubernetes托管版集群.

- **VPC**: Select the VPC created in Step 1: Plan the cluster network.
- **VSwitch**: Select the vSwitch created in Step 1: Plan the cluster network.
- **Network Plug-in**: Select **Terway**.
  If you set **Network Plug-in** to **Terway**, you must set **Terway Mode**.
  - Select or clear **Assign One ENI to Each Pod**.
    - If you select the check box, an ENI is assigned to each pod.
    - If you clear the check box, an ENI is shared among multiple pods. A secondary IP address that is provided by the ENI is assigned to each pod.

      Note To select the **Assign One ENI to Each Pod** check box, you must submit a ticket to apply to be added to a whitelist.
  - Select or clear **IPVLAN**.
    - This option is available only when you clear Assign One ENI to Each Pod.
    - If you select IPVLAN, IPVLAN and extended Berkeley Packet Filter (eBPF) are used for network virtualization when an ENI is shared among multiple pods. This improves network performance. Only the Alibaba Cloud Linux 2 operating system is supported.
    - If you clear IPVLAN, policy-based routes are used for network virtualization when an ENI is shared among multiple pods. The CentOS 7 and Alibaba Cloud Linux 2 operating systems are supported. This is the default setting.

    For more information about the IPVLAN feature in Terway mode, see Terway IPvlan.

- Select or clear **Support for NetworkPolicy**.
  - The NetworkPolicy feature is available only when you clear Assign One ENI to Each Pod. By default, Assign One ENI to Each Pod is unselected.
  - If you select Support for NetworkPolicy, you can use Kubernetes network policies to control the communication among pods.
  - If you clear Support for NetworkPolicy, you cannot use Kubernetes network policies to control the communication among pods. This prevents Kubernetes network policies from overloading the Kubernetes API server.

- **Pod VSwitch**: Select the pod vSwitch created in Step 1: Plan the cluster network.
- **Service CIDR**: Use the default settings.

Terway IPvlan

If you select the Terway network plug-in when you create a cluster, you can choose to enable the Terway IPvlan mode. The Terway IPvlan mode provides high-performance networks for pods and Services based on IPvlan and Extended Berkeley Packet Filter (eBPF) technologies.

Compared with the default Terway mode, the Terway IPvlan mode optimizes the performance of pod networks, Service networks, and network policies.

- Pod networks are directly implemented based on the sub-interfaces of elastic network interfaces (ENIs) in IPvlan L2 mode. This significantly simplifies network forwarding on the host and reduces the latency by 30% compared with the traditional mode. The performance of pod networks is almost the same as that of the host network.
- Service networks are implemented based on the eBPF technology instead of the kube-proxy mode. Traffic forwarding no longer depends on the host iptables or IP Virtual Server (IPVS). This maintains almost the same performance in larger-scale clusters and offers better scalability. Compared with traffic forwarding based on IPVS and iptables, this new approach greatly reduces network latency in scenarios that involve a large number of new connections and port reuse.
- The network policies of pods are implemented based on the eBPF technology instead of iptables. This way, large numbers of iptables rules are no longer generated on the host and the impact of network policies on network performance is minimized.

Limits of the Terway IPvlan mode

- Only the Alibaba Cloud Linux 2 operating system is supported.
- The Sandboxed-Container runtime is not supported.
- The implementation of network policies is different from when the default Terway mode is used.
  - The CIDR selector has a lower priority than the pod selector. Additional pod selectors are required if the CIDR block of pods is within the CIDR range specified by the CIDR selector.
  - The except keyword of the CIDR selector is not fully supported. We recommend that you do not use the except keyword.
  - If you use a network policy of the Egress type, you cannot access pods in the host network or the IP addresses of nodes in the cluster.
- You may fail to access the Internet-facing SLB instance that is associated with a LoadBalancer Service from within the cluster due to loopback issues. For more information, see Why am I unable to access an SLB instance?.

Scenarios

- Middleware and microservices
  Avoids performance degradation in large-scale deployments and reduces the network latency of microservices.

- Gaming and live streaming applications
  Significantly reduces network latency and resource competition among multiple instances.
- High-performance computing
  High-performance computing requires high-throughput networks. The Terway IPvlan mode reduces CPU overhead and saves computing resources for core workloads.

Related information

- Overview
- Plan CIDR blocks for an ACK cluster
- Create vSwitches for an ACK cluster that has Terway installed

6.3.2. Create vSwitches for an ACK cluster that has Terway installed

When the IP addresses provided by a vSwitch in a Container Service for Kubernetes (ACK) cluster are exhausted, you must create vSwitches in the virtual private cloud (VPC) where the cluster is deployed. This topic describes how to expand the IP address space of an ACK cluster by creating vSwitches.

Symptoms of insufficient IP addresses

In a cluster that has Terway installed, the following symptoms indicate that the IP addresses provided by the vSwitch in the cluster are exhausted:

- You fail to create pods. The pod that you want to create stays in the ContainerCreating state. In this case, run the following command to query the Terway log on the node where the pod is deployed:

  ```
  kubectl logs --tail=100 -f terway-eniip-zwjwx -n kube-system -c terway
  ```

  If an error message similar to the following content is returned, it indicates that the IP addresses of the vSwitch are exhausted. The pod cannot be created and stays in the ContainerCreating state because no IP address is available.

  ```
  time="2020-03-17T07:03:40Z" level=warning msg="Assign private ip address failed: Aliyun API Error: RequestId: 2095E971-E473-4BA0-853F-0C41CF52651D Status Code: 403 Code: InvalidVSwitchId.IpNotEnough Message: The specified VSwitch \"vsw-AAA\" has not enough IpAddress., retrying"
  ```

- Log on to the VPC console. In the left-side navigation pane, click **VSwitches**. On the VSwitches page, find the vSwitch and verify that 0 is displayed in the **Number of Available Private IPs** column.

Create a vSwitch

To create a vSwitch, perform the following steps:

1. Log on to the VPC console and create a vSwitch. You must create the vSwitch in the same region as the preceding one whose IP addresses are exhausted. For more information about how to create a vSwitch, see Work with vSwitches.

   > **Note** To provide sufficient IP addresses for an increasing number of pods in the ACK cluster, we recommend that the CIDR block of the vSwitch contain at least 8,192 IP addresses. This means that the network bits of the CIDR block must be no greater than 19 in length.

2. Run the following command to add the newly created vSwitch to the ConfigMap of Terway:

   ```
   kubectl edit cm eni-config -n kube-system
   ```

   The following code provides an example on how to configure the ConfigMap:

```
eni_conf: |
{
"version": "1",
"max_pool_size": 25,
"min_pool_size": 10,
"vswitches": {"cn-shanghai-f":["vsw-AAA", "vsw-BBB"]},
"service_cidr": "172.21.0.0/20",
"security_group": "sg-CCC"
}
```

In this example, `vsw-BBB` is added to the value of `vswitches`. `vsw-AAA` represents the existing vSwitch that has insufficient IP addresses.

3. (Optional)Upgrade Terway.

   Upgrade Terway to the latest version. If Terway is already upgraded to the latest version, skip this step.

   i. Log on to the ACK console.

   ii. In the left-side navigation pane of the ACK console, click **Clusters**.

   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

   iv. In the left-side navigation pane, choose **Operations > Add-ons**. On the **Add-ons** page, click the **Networking** tab.

   v. On the **Networking** tab, find the Terway plug-in and click **Upgrade**.

4. Run the following commands to delete all pods that use Terway. ACK automatically recreates the pods.

   - If an elastic network interface (ENI) is shared among multiple pods, run the following command:

     ```
     kubectl delete -n kube-system pod -l app=terway-eniip
     ```

   - If an ENI is exclusive to one pod, run the following command:

     ```
     kubectl delete -n kube-system pod -l app=terway-eni
     ```

   Run the following command to check whether all pods are recreated:

   ```
   kubectl get pod -n kube-system | grep terway
   ```

5. After the preceding steps are completed, you can create a pod to check whether the pod is assigned an IP address from the newly created vSwitch.

If exceptions occur in the preceding steps, Submit a ticket.

Related information

- Use the Terway plug-in
- 使用网络策略Network Policy

6.3.3. Add a vSwitch to a cluster based on a secondary CIDR block

When you create a cluster in Container Service for Kubernetes (ACK), you must specify a virtual private cloud (VPC) for the cluster. If you want to expand the cluster, you must deploy cloud resources in the VPC to which the cluster belongs. If the CIDR block of the VPC does not have sufficient IP addresses, you can add a secondary CIDR block to the VPC. This way, you can expand the cluster based on your business requirements. This topic describes how to add a vSwitch to a cluster based on a secondary CIDR block.

Prerequisites

- The CIDR block of the VPC to which the cluster belongs does not have sufficient IP addresses.
- A managed ACK cluster is created in February 2021 or later or a dedicated ACK cluster is created. For more information, see 创建Kubernetes托管版集群 or Create a dedicated Kubernetes cluster.

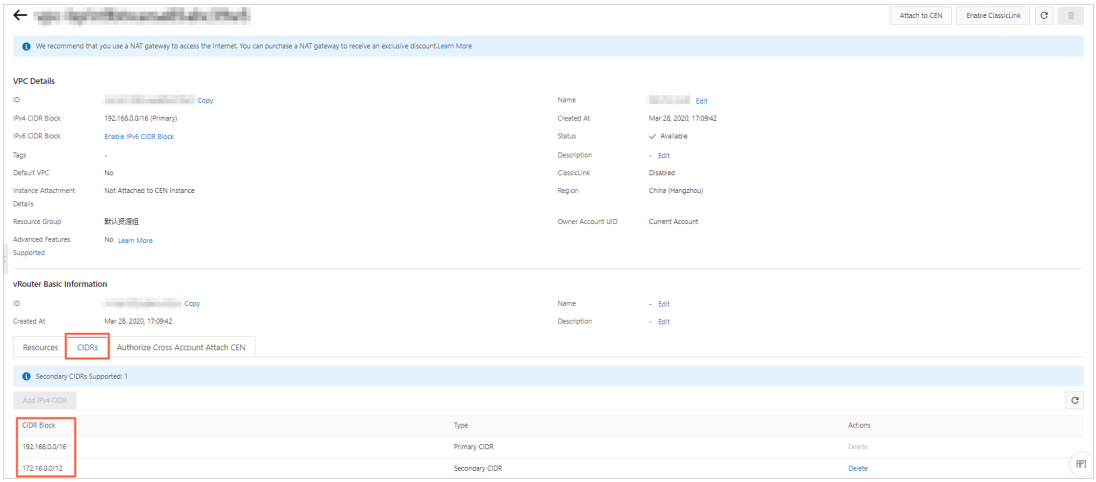
> **Notice** You cannot add a secondary CIDR block to the VPC if your managed ACK cluster was created earlier than February 2021. In this case, you can migrate workloads to a professional managed cluster and then add a vSwitch to the cluster based on a secondary CIDR block. For more information about how to migrate workloads from standard managed clusters to professional managed clusters, see 热迁移ACK标准托管集群至Pro托管集群.

Procedure

1. Select an available CIDR block.
   i. Check the CIDR blocks that are in use.

   The CIDR blocks include but are not limited to:

   - The current CIDR block of the VPC.
     For more information about how to check the current CIDR block of a VPC, see View a VPC.

   - The CIDR blocks of the pods and Services that are deployed in the VPC.
     For more information about how to check the CIDR blocks of pods and Services, see View basic information.

     > **Note**
     > - If the cluster uses the Terway network plug-in, check the CIDR block of Services.
     > - If the cluster uses the Flannel network plug-in, check the CIDR blocks of pods and Services.

   - The CIDR blocks of connections over Express Connect circuits, VPN gateways, and Cloud Enterprise Network (CEN) instances that are connected to the VPC.

ii. Select a CIDR block that does not overlap with the preceding CIDR blocks, and use this CIDR block as the secondary CIDR block of the VPC.

For example, a cluster that uses the Flannel network plug-in may use the following CIDR blocks:

- VPC CIDR block: 192.168.0.0/16
- Pod CIDR block: 172.20.0.0/16
- Service CIDR block: 172.21.0.0/16
- The VPC is not connected with connections over Express Connect circuits, VPN gateways, or Cloud Enterprise Network (CEN) instances.

In this case, you can use 10.0.0.0/8 as a secondary CIDR block.

2. Add a secondary CIDR block and vSwitch in the VPC console.
   i. Log on to the VPC console.
   ii. In the top navigation bar, select the region where the VPC is deployed.
   iii. On the **VPC** page, find the VPC that you want to manage and click its ID.
   iv. On the **VPC Details** page, click the **CIDRs** tab. Then, click **Add IPv4 CIDR** and enter the IPv4 CIDR block that you selected in the previous step.
   v. You can create a vSwitch in the secondary CIDR block based on your business requirements. The zone of the vSwitch must be the same as the zone of the secondary CIDR block.

   For more information about how to create a vSwitch, see Create a vSwitch.
3. Add rules to the security group of the cluster to allow inbound and outbound access to the secondary CIDR block over all protocols.

   For more information about how to add security group rules, see Add security group rules.
4. Expand the cluster by creating nodes in the new vSwitch.

   When you expand the cluster, you must deploy new nodes in the new vSwitch. The CIDR block of the new vSwitch belongs to the secondary CIDR block of the VPC. For more information, see Expand an ACK cluster.

   > **Notice** If the cluster uses the Terway network plug-in, you can modify the vSwitch settings of the Terway plug-in to provide more IP addresses for pods. For more information about how to enable pods to use the IP addresses provided by the new vSwitch, see Create vSwitches for an ACK cluster that has Terway installed.

   After new nodes are deployed in the new vSwitch,
   - the master nodes and new nodes can directly communicate with each other in dedicated ACK clusters. No extra configuration is required.
   - extra configuration is required on the control planes of standard managed clusters and professional managed clusters. Otherwise, you cannot use the API server to run **kubectl exec** or **kubectl logs** commands on the pods of new nodes. You must Submit a ticket to ask technical support staff to reconfigure the control plane.
5. Verify that the secondary CIDR block of the VPC works as expected.

   Perform the following steps:
   - Verify that the IP addresses of new nodes and the IP addresses of the pods hosted by new nodes belong to the secondary CIDR block of the VPC.
   - Verify that the new nodes are in the **Ready** state.
   - Verify that the new nodes can communicate with the existing nodes in the cluster and that the pods hosted on new nodes can communicate with the pods hosted on existing nodes.

Related information

- Plan CIDR blocks for an ACK cluster
- Add a secondary IPv4 CIDR block
- Create vSwitches for an ACK cluster that has Terway installed
- 热迁移ACK标准托管集群至Pro托管集群

6.3.4. Use the host network

If a pod runs in the host network of the node where the pod is deployed, the pod can use the network namespace and network resources of the node. In this case, the pod can access loopback devices, listen to addresses, and monitor the traffic of other pods on the node.

Prerequisites

- A Container Service for Kubernetes (ACK) cluster is created. For more information, see Create a dedicated Kubernetes cluster.
- The ACK cluster can be accessed with kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

Procedure

1. Open the YAML file *host-network.yaml* and set `hostNetwork` to true under *spec*.

   The following example shows the YAML file:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
    name: nginx
   spec:
    hostNetwork: true
    containers:
    - name: nginx
      image: nginx
   ```

2. Run the following command to create a pod:

   ```
   kubectl apply -f host-network.yaml
   ```

3. Run the following command to check whether the pod runs in the host network of the node where the pod is deployed:

   ```
   kubectl get pod -o wide
   ```

   The command output shows that the IP address of the pod is the same as that of the node. This indicates that the pod runs in the host network of the node.

   ```
   NAME  READY  STATUS   RESTARTS  AGE  IP          NODE              NOMINATED NODE
   nginx 1/1    Running  0         29s  192.168.XX.XX cn-zhangjiakou.192.168.XX.XX <none>
   ```

6.3.5. Associate an EIP with a pod

In most cases, a pod uses a private IP address. In a Container Service for Kubernetes (ACK) cluster that uses Terway, a pod may require a dedicated public IP address in some cases. This topic describes how to associate an elastic IP address (EIP) with a pod in a cluster that uses Terway.

Context

In most cases, to enable a pod to access the Internet, the IP address of the pod is converted to an EIP that is associated with the host or the Network Address Translation (NAT) gateway of the virtual private cloud (VPC) based on Source Network Address Translation (SNAT) rules. The inbound Internet traffic to the pod is routed through the related LoadBalancer type Services. A pod may require a dedicated public IP address in the following scenarios.

- The pod exposes random ports to the Internet. For example, game servers and audio conferences that use the User Datagram Protocol (UDP). For example, if the Real Time Streaming Protocol (RTSP) is used, different ports are used for different clients.
- The pod competes with other pods for a public IP address to access the Internet. In this case, the pod requires a dedicated public IP address to access the Internet.

For more information about how to use the Terway network plug-in in an ACK cluster, see Use the Terway plug-in.

Step 1: Upgrade Terway to the latest version

Upgrade Terway in your ACK cluster to a version that supports EIP. For more information about how to upgrade Terway, see Manage system components.

> **Note** Terway `V1.0.10.280-gdc2cb6c-aliyun` and later support EIP. We recommend that you upgrade Terway to the latest version.

Step 2: Configure and grant permissions to manage EIPs

You must configure and grant EIP-related permissions to the cluster that uses Terway.

1. Grant EIP-related permissions to the Resource Access Management (RAM) role of the cluster that uses Terway.

   ### For managed Kubernetes clusters that were created before June 2020 or dedicated Kubernetes clusters, you must grant permissions to the worker RAM role of the cluster.

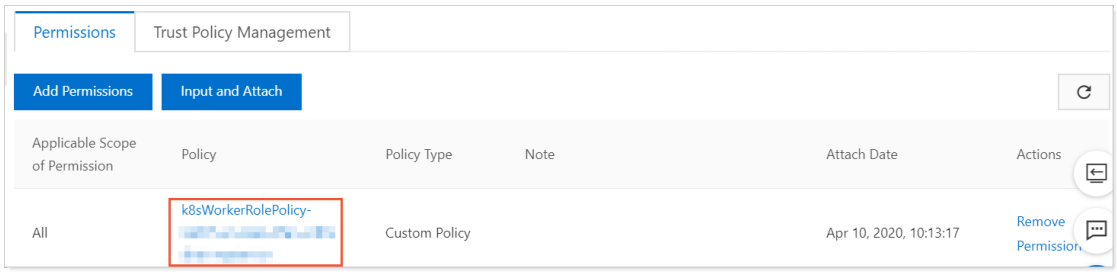
   i. Log on to the ACK console.

   ii. In the left-side navigation pane of the ACK console, click **Clusters**.

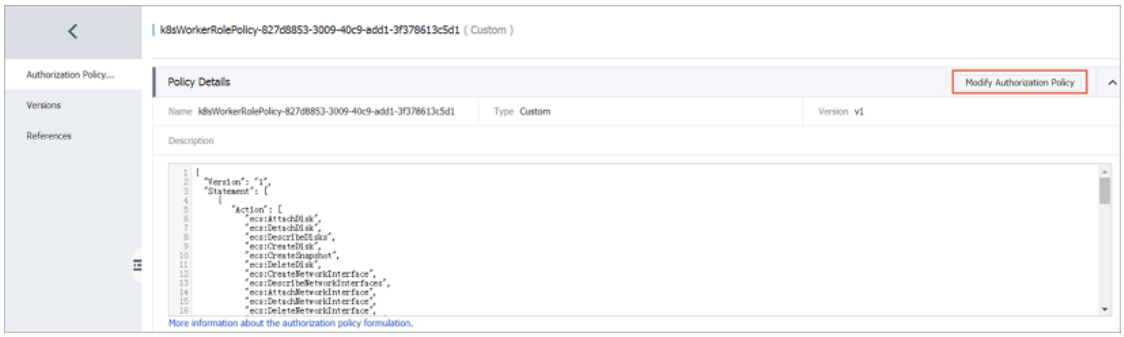
   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

   iv. Click the **Cluster Resources** tab and click the hyperlink on the right side of **Worker RAM Role**.

   v. Click the **Permissions** tab and click the name of the policy. The details page of the policy appears.



vi. Click **Modify Policy Document**. In the **Modify Policy Document** panel that appears on the right side of the page, add the following content to the `Action` field in the **Policy Document** section. Click **OK**.




```
"vpc:DescribeVSwitches",
"vpc:AllocateEipAddress",
"vpc:DescribeEipAddresses",
"vpc:AssociateEipAddress",
"vpc:UnassociateEipAddress",
"vpc:ReleaseEipAddress"
```

> **Note** Add a comma (,) to the end of each line in the `Action` field.

Grant the following EIP-related permissions to the AliyunCSManagedNetworkRole RAM role in the RAM console.

```
"vpc:DescribeVSwitches",
"vpc:AllocateEipAddress",
"vpc:DescribeEipAddresses",
"vpc:AssociateEipAddress",
"vpc:UnassociateEipAddress",
"vpc:ReleaseEipAddress"
```

For standard managed Kubernetes clusters that were created in and after June 2020 or professional managed Kubernetes clusters, you must grant permissions to the AliyunCSManagedNetworkRole RAM role.

i. On the RAM role AliyunCSManagedNetworkRole page, find the policy and click **Add Permissions** on the right side of the page.

ii. In the **Add Permissions** panel, click **+ Create Policy** in the **Select Policy** section to create a custom policy.

For more information about how to create a custom policy, see Create a custom policy.

iii. Add the following policy content to the custom policy.

For more information about how to modify a custom policy, see Modify a custom policy.

```
"vpc:DescribeVSwitches",
"vpc:AllocateEipAddress",
"vpc:DescribeEipAddresses",
"vpc:AssociateEipAddress",
"vpc:UnassociateEipAddress",
"vpc:ReleaseEipAddress"
```

2. Configure Terway to support EIP.
   i. Run the following command to modify the eni_conf ConfigMap of the Terway configurations:

   ```
   kubectl edit cm eni-config -n kube-system
   ```

   ii. Add the following content to the eni_conf ConfigMap:

   ```
   "enable_eip_pool": "true"
   ```

   > **Notice** If you want to disassociate the existing EIP when you associate a new one, add "allow_eip_rob": "true" to the eni_conf ConfigMap.

   iii. Run the following command to redeploy Terway:

   ```
   kubectl delete pod -n kube-system -l app=terway-eniip
   ```

Grant EIP-related permissions to the RAM role of the cluster that uses Terway

Step 3: Add annotations to create and associate an EIP with a pod

You can add annotations to the configurations of a pod to create or associate an EIP with the pod. Add annotations as described in the following scenarios:

- **Automatically create and associate an EIP with a pod**
  - Add the following annotation to create and associate an EIP with a pod:

    ```
    k8s.aliyun.com/pod-with-eip: "true"
    ```

    The following YAML file is provided as an example.

    ```
    apiVersion: apps/v1
    kind: Deployment
    metadata:
     name: nginx-deployment-basic
     labels:
       app: nginx
    spec:
     replicas: 2
     selector:
       matchLabels:
         app: nginx
     template:
       metadata:
         annotations:
           k8s.aliyun.com/pod-with-eip: "true" # Specifies that an EIP is automatically created and associated with t
    he NGINX containers.
         labels:
           app: nginx
       spec:
         containers:
         - name: nginx
           image: nginx
    ```

- Add the following annotation to specify the bandwidth for the EIP. The default EIP bandwidth is 5 Mbit/s.

  ```
  k8s.aliyun.com/eip-bandwidth: "5"
  ```

- **Specify an existing EIP**
  - Add the following annotation to specify the ID of an existing EIP and associate it with a pod:

    ```
    k8s.aliyun.com/pod-eip-instanceid: "<youreipInstanceId>"
    ```

    > **Note**
    > - An EIP cannot be associated with multiple pods. Therefore, you cannot use this annotation for pods of Deployments and StatefulSets.
    > - By default, if an EIP is already associated with a pod, you fail to create a new EIP for the pod. If you want to disassociate the existing EIP and create a new one, add the "allow_eip_rob": "true" setting to the eni_conf ConfigMap as described in the preceding step.
    > - You can specify an existing EIP only when the application runs one pod replica. For example, a Deployment that runs only one pod replica.

Verify the configurations

When the pod changes to the Running state, you can check the value of k8s.aliyun.com/allocated-eipAddress in the pod annotations. The value is the associated EIP. You can access the pod through the EIP.

Related information

- Plan CIDR blocks for an ACK cluster
- Use the Terway plug-in

6.3.6. Configure multiple route tables for a VPC

Container Service for Kubernetes (ACK) uses Cloud Controller Manager (CCM) to add route entries to the route table of the virtual private cloud (VPC) where the cluster is deployed. This enables network connections between pods in the cluster. You can update the *cloud-config* file to configure multiple route tables for the VPC where a dedicated ACK cluster is deployed. This topic describes how to configure multiple route tables for the VPC where a dedicated ACK cluster is deployed.

Prerequisites

- A dedicated ACK cluster is created. For more information, see Create a dedicated Kubernetes cluster.
- The Flannel network plug-in is used in the dedicated ACK cluster.
- The CCM version is later than v1.9.3.86-g4454991-aliyun. For more information about how to check the CCM version, see Manage system components.

Context

When multiple route tables are configured for a VPC, you can select to associate a route table to Elastic Compute Service (ECS) instances. Previous versions of CCM allow you to configure only one route table for a VPC. Therefore, you must upgrade the CCM component to the latest version in the ACK console. For more information about how to upgrade the CCM version, see Manage system components.

> **Note** Only a dedicated ACK cluster allows you to configure multiple route tables for the VPC where the cluster is deployed. To configure multiple route tables for the VPC where a managed ACK cluster is deployed, Submit a ticket.

Configure multiple route tables for the VPC by using a kubectl client

To use a kubectl client to configure multiple route tables for the VPC, make sure that the kubectl client is connected to the dedicated ACK cluster. For more information, see Connect to Kubernetes clusters by using kubectl.

1. Update the *cloud-config* file.

   Run the `kubectl edit cm -n kube-system cloud-config` command to edit the *cloud-config* file. Replace `${ROUTE_TABLES_IDS}` with the IDs of route tables in the VPC. Make sure that the ID of the system route table is included. Separate multiple route table IDs with commas (,). For example, vtb-t4n788888****,vtb-t4n7k6u3m0n840799****.
   The following *cloud-config* file is provided as an example.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
    name: cloud-config
    namespace: kube-system
   data:
    cloud-config.conf: |-
     {
       "Global": {
         "routeTableIDs": "${ROUTE_TABLES_IDS}"
       }
     }
   ```

2. Run the following command to restart the pod that runs CCM.

   ```
   kubectl -n kube-system delete po -lapp=cloud-controller-manager
   ```

   After the pod is restarted, verify that route entries of the cluster nodes are displayed in the specified route tables.

Configure multiple route tables for the VPC by using the ACK console

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
5. Select the kube-system namespace. Find the *cloud-config* ConfigMap and click **Edit YAML** in the Actions column.
6. In the **View in YAML** panel, set routeTableIDs to the IDs of route tables in the VPC. Make sure that the ID of the system route table is included. Separate multiple route table IDs with commas (,). For example, vtb-t4n788888****,vtb-t4n7k6u3m0n840799****. Then, click **OK**.

   ```
   1  apiVersion: v1
   2  data:
   3    cloud-config.conf: |-
   4      {
   5        "Global": {
   6
   7          "routeTableIDs": "",
   8          "clusterID": "cb2eda2
   9        }
   10     }
   ```

7. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

8. In the kube-system namespace, find the alicloud-application-controller Deployment and choose **More > Redeploy** in the Actions column.
   After the redeployment is completed, verify that route entries of the cluster nodes are displayed in the specified route tables.

6.3.7. Enable an existing cluster to access the Internet by using SNAT

If an Elastic Compute Service (ECS) instance or an ACK cluster does not have a public IP address, you can create an SNAT entry in the virtual private cloud (VPC) where the ECS instance or ACK cluster is deployed to provide a proxy to enable access to the Internet. If Source Network Address Translation (SNAT) is not enabled when you create a Container Service for Kubernetes (ACK) cluster, you can enable SNAT in the ACK console after the cluster is created. This topic describes how to enable SNAT for existing ACK clusters in the ACK console. SNAT allows existing ACK clusters to access the Internet.

Context

You cannot call API operations to enable SNAT for existing clusters. For more information about SNAT, see What is NAT Gateway?.

The following flowchart shows the steps to enable SNAT for an existing ACK cluster to access the Internet.

Procedure

1. Create a NAT gateway.
   i. Log on to the NAT Gateway console.
   ii. In the left-side navigation pane, click **NAT Gateway**.
   iii. On the **NAT Gateway** page, click **Create NAT Gateway**.
   For more information about the parameters required to create a NAT gateway, see Create NAT gateways.

   > **Note** The NAT gateway must be created in the same region and VPC as the ACK cluster.

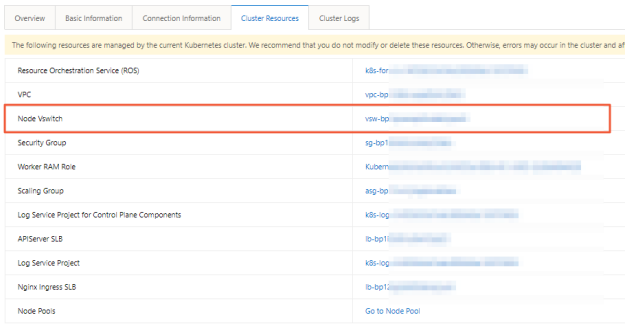
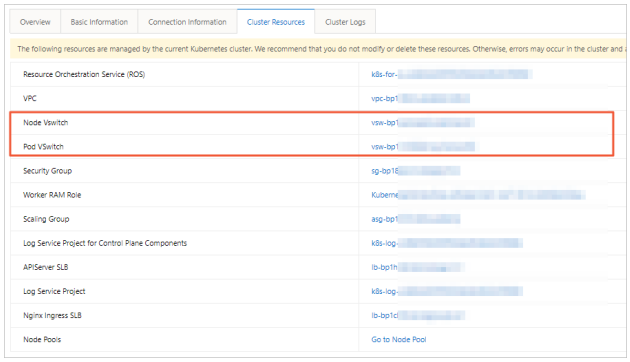
2. Create an EIP.
   In the left-side navigation pane, choose **Elastic IP Addresses > Elastic IP Addresses**. On the **Elastic IP Addresses** page, click **Create EIP**.
   If you have an elastic IP address (EIP), skip this step.
3. Associate the EIP with the created NAT gateway.
   i. On the **NAT Gateway** page, find the newly created NAT gateway and choose > **Bind Elastic IP Address** in the Actions column.
   ii. In the **Associate EIP** dialog box, select a resource group from the **Resource Group** drop-down list and select the EIP that you created from the **Select Existing EIPs** drop-down list.
   iii. Click **OK**.
4. Create an SNAT entry for the NAT gateway.

i. On the **NAT Gateway** page, find the newly created NAT gateway and click **Manage** in the Actions column.

ii. On the **SNAT Management** tab, click **Create SNAT Entry**.

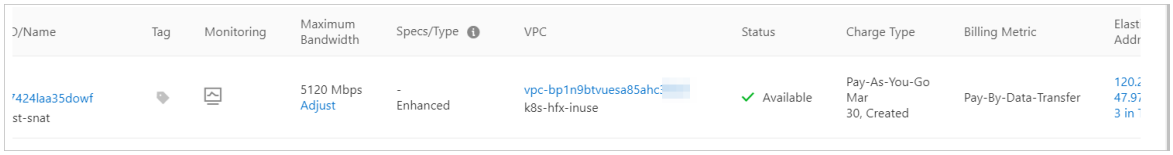
iii. On the **Create SNAT Entry** page, set the parameters as described in the following table and click **Confirm**.

For more information about the parameters, see Create a SNAT entry.

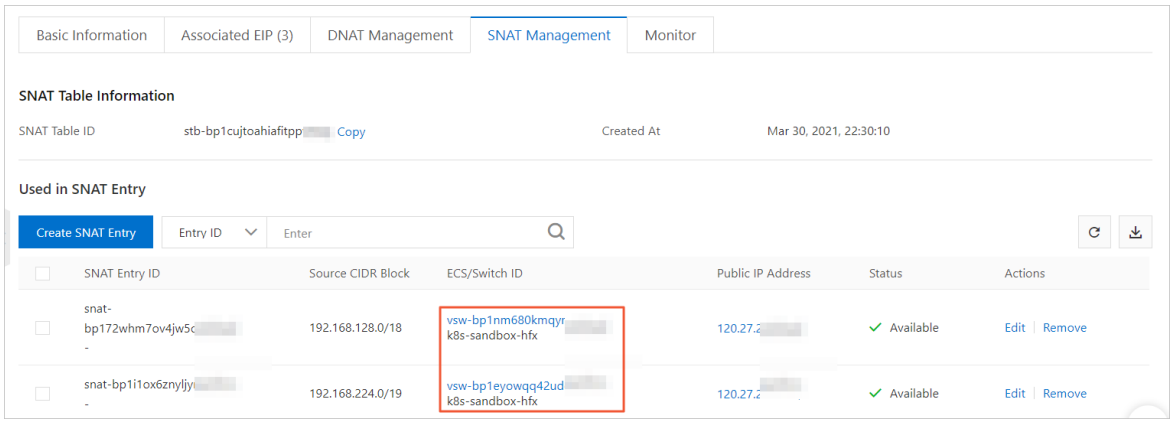
Parameter	Description
SNAT Entry	Select **Specify VSwitch** and select the vSwitches that are used by the cluster. ■ If the cluster uses Terway as the network plug-in, select the node vSwitches and pod vSwitches. ■ If the cluster uses Flannel as the network plug-in, select the node vSwitches. Perform the following steps to check the IDs of the vSwitches used by the cluster: a. Log on to the ACK console. b. In the left-side navigation pane, click **Clusters**. On the Clusters page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column. c. On the details page of the cluster, click the **Cluster Resources** tab to view the vSwitch IDs. Clusters that use Flannel  Clusters that use Terway 
Select Public IP Address	Select a public IP address that is used to access the Internet.

After the SNAT entry is created and SNAT rules are configured, SNAT is enabled for the cluster. You can

log on to the NAT Gateway console to view the details of the NAT gateway, such as the EIP used by SNAT. The following figure shows a NAT gateway that is used for an ACK cluster that uses Terway as the network plug-in. This NAT gateway is configured with SNAT rules to enable the cluster to access the Internet.

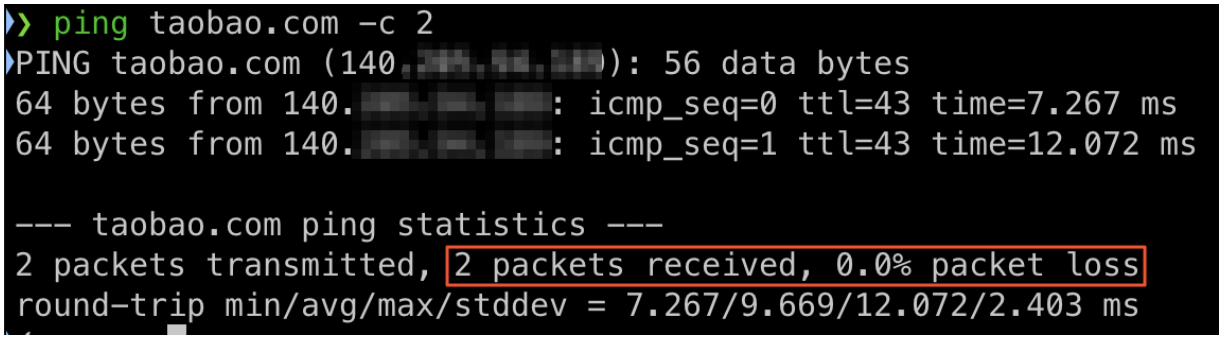


Click the name of the NAT gateway. On the **SNAT Management** tab of the details page, you can check whether the public IP address is associated with the vSwitches used by the cluster. The following figure shows the SNAT entries created for the cluster that uses Terway as the network plug-in.



Result

Log on to a node of the cluster and access the Internet to verify that the node can access the Internet and no packet loss occurs during data transmission.



6.3.8. FAQ about container networks

This topic provides answers to some frequently asked questions about container networks.

- How do I resolve the issue that Flannel becomes incompatible with clusters of Kubernetes 1.16 or later after I manually upgrade Flannel?
- How do I resolve the issue that a pod is not immediately ready for communication after it is started?
- How do I enable a pod to access the Service that is exposed on it?
- Which network plug-in should I choose for an ACK cluster, Terway or Flannel?
- How do I plan the network of a cluster?
- Can I use hostPorts to create port mappings in an ACK cluster?
- Can I configure multiple route tables for the VPC where my cluster is deployed?

- How do I check the network type and vSwitches of the cluster?
- How do I check the cloud resources used in an ACK cluster?

How do I solve the issue that Flannel becomes incompatible with clusters of Kubernetes 1.16 or later after I manually upgrade Flannel?

Symptom:

After a cluster is upgraded to v1.16 or later, the states of nodes in the cluster change to NotReady.

Cause:

You manually upgraded Flannel without updating the Flannel configuration. As a result, kubelet cannot recognize Flannel.

Solution:

1. Edit the Flannel configuration file to add the `cniVersion` field.

   ```
   kubectl edit cm kube-flannel-cfg -n kube-system
   ```

   Add the `cniVersion` field based on the following example:

   ```
   "name": "cb0",
   "cniVersion":"0.3.0",
   "type": "flannel",
   ```

2. Restart Flannel.

   ```
   kubectl delete pod -n kube-system -l app=flannel
   ```

How do I resolve the issue that a pod is not immediately ready for communication after it is started?

Symptom:

After a pod is started, you must wait for a period of time before the pod is ready for communication.

Cause:

It requires a period of time for network policies to take effect. To resolve this issue, you can disable network policies.

Solution:

1. Edit the ConfigMap of Terway to add settings to disable network policies.

   ```
   kubectl edit cm -n kube-system eni-config
   ```

   Add the following field to the ConfigMap:

   ```
   disable_network_policy: "true"
   ```

2. (Optional)If Terway is not upgraded to the latest version, log on to the Container Service for Kubernetes (ACK) console and upgrade Terway.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
   iv. In the left-side navigation pane, choose **Operations > Add-ons**.
   v. On the **Add-ons** page, click the **Networking** tab, find the Terway section, and then click **Upgrade**.

vi. In the **Note** message, click **OK**.

3. Restart all of the pods that use Terway.

```
kubectl delete pod -n kube-system -l app=terway-eniip
```

How do I enable a pod to access the Service that is exposed on it?

Issue:

Pods are not allowed to access Services that are exposed on them. When a pod accesses the Service that is exposed on it, the performance of the Service becomes unstable or scheduling errors may occur.

Cause:

By default, Flannel does not allow loopback requests.

Solution:

- Use a headless Service to expose and access Services. For more information, see Headless Services.

  > **Note** We recommend that you use this method.

- Recreate a cluster that uses the Terway network plug-in. For more information, see Use the Terway plug-in.
- Modify the Flannel configuration, reinstall Flannel, and recreate the pod.

  > **Note** We recommend that you do not use this method because the configuration of Flannel may be overwritten if Flannel is upgraded.

  i. Add `hairpinMode: true` to cni-config.json.

  ```
  kubectl edit cm kube-flannel-cfg -n kube-system
  ```

  Add the following field to the configuration:

  ```
  hairpinMode: true
  ```

  ii. Restart Flannel.

  ```
  kubectl delete pod -n kube-system -l app=flannel
  ```

  iii. Delete and recreate the pod.

Which network plug-in should I choose for an ACK cluster, Terway or Flannel?

The following introduction describes the Flannel and Terway network plug-ins for ACK clusters.

You can select one of the following network plug-ins when you create an ACK cluster:

- Flannel: a simple and stable Container Network Interface (CNI) plug-in developed by the Kubernetes community. You can use Flannel with Virtual Private Cloud (VPC) of Alibaba Cloud. This ensures that your clusters and containers run in a high-speed and stable network. However, Flannel provides only basic features and does not support standard Kubernetes network policies.
- Terway: a network plug-in developed by ACK. Terway provides all the features of Flannel and allows you to attach Alibaba Cloud elastic network interfaces (ENIs) to containers. You can use Terway to define access control policies based on standard Kubernetes network policies for intercommunication among containers. Terway also supports bandwidth throttling on individual containers. If you do not want to use Kubernetes network policies, you can choose Flannel. In other cases, we recommend that you choose

Terway. For more information about Terway, see Use the Terway plug-in.

How do I plan the network of a cluster?

When you create an ACK cluster, you must specify a VPC, vSwitches, CIDR blocks of pods, and CIDR blocks of Services. We recommend that you plan the IP address of each Elastic Compute Service (ECS) instance in the cluster, the CIDR blocks of Kubernetes pods, and the CIDR blocks of Services before you create an ACK cluster. For more information, see Plan CIDR blocks for an ACK cluster.

Can I use hostPorts to create port mappings in an ACK cluster?

- No. You cannot use hostPorts to create port mappings in an ACK cluster.
- A pod in a VPC can be accessed by other cloud resources that are deployed in the same VPC through the endpoint of the pod in the VPC. Therefore, port mapping is not required.
- Only Services of the NodePort and LoadBalancer types can be exposed to the Internet.

Can I configure multiple route tables for the VPC where my cluster is deployed?

Only dedicated ACK clusters allow you to configure multiple route tables for the VPC. For more information, see Configure multiple route tables for a VPC. To configure multiple route tables for the VPC where a managed ACK cluster is deployed, Submit a ticket.

How do I check the network type and vSwitches of the cluster?

ACK supports two types of container network: Flannel and Terway.

To check the network type of the cluster, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the Clusters page, find the cluster that you want to manage and click the name of the cluster, or click **Details** in the Actions column. The details page of the cluster appears.
4. On the **Basic Information** tab, check the value of **Network Plug-in** in the **Cluster Information** section.
   - If the value of **Network Plug-in** is **terway-eniip**, it indicates that the Terway network is used.
   - If the value of **Network Plug-in** is **Flannel**, it indicates that the Flannel network is used.

To check the vSwitches used in the container network, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the Clusters page, find the cluster that you want to manage and click the name of the cluster, or click **Details** in the Actions column. The details page of the cluster appears.
4. On the **Cluster Resources** tab, check the vSwitches used in the network.
   - If the cluster uses the Terway network, you can find information about **Node Vswitch** and **Pod Vswitch** on the **Cluster Resources** tab.
   - If the cluster uses the Flannel network, you can find information about **Node Vswitch** on the **Cluster Resources** tab.

How do I check the cloud resources used in an ACK cluster?

You can perform the following steps to check the cloud resources used in ACK clusters, such as vSwitches, VPCs, and worker RAM roles.

1. Log on to the ACK console.

2. In the left-side navigation pane, click **Clusters**.
3. On the Clusters page, find the cluster that you want to manage and click the name of the cluster, or click **Details** in the Actions column. The details page of the cluster appears.
4. On the **Cluster Resources** tab, check information about the cloud resources used in the cluster.

6.4. Service Management

6.4.1. Considerations for configuring a LoadBalancer type Service

When you specify `Type=LoadBalancer` for a Service, Cloud Controller Manager (CCM) creates and configures Server Load Balancer (SLB) resources for the Service, including SLB instances, listeners, and VServer groups. This topic describes the considerations for configuring a LoadBalancer type Service and the policies that are used by CCM to update SLB resources.

Policies that are used by CCM to update SLB resources

Container Service for Kubernetes (ACK) allows you to specify an existing SLB instance for a Service. You can also use CCM to automatically create one for the Service. The two methods use different policies to update SLB resources. The following table describes the differences.

Resource object	Existing SLB instance	SLB instance created and managed by CCM
SLB	Use the following annotation to specify an existing SLB instance for a Service: `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id` • CCM uses the specified SLB instance to enable load balancing. You can use other annotations to configure the SLB instance. CCM automatically creates VServer groups for the instance. • When the Service is deleted, CCM does not delete the existing SLB instance that is specified in the annotation.	• CCM automatically creates, configures, and manages SLB resources based on the Service configuration, including the SLB instance, listeners, and VServer groups. • When the Service is deleted, CCM deletes the created SLB instance.
Listener	Use the following annotation to configure listeners: `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners:` . • If you set the annotation to false, CCM does not configure or manage listeners for the SLB instance. • If you set the annotation to true, CCM configures and manages listeners for the SLB instance based on the Service configuration. If the SLB instance has existing listeners, CCM creates new listeners to replace the existing ones.	CCM configures listeners for the SLB instance based on the Service configuration.

Resource object	Existing SLB instance	SLB instance created and managed by CCM

VServer group	When the endpoint of an Elastic Compute Service (ECS) instance in a VServer group for a Service changes or the cluster nodes are changed, CCM updates the VServer groups.
	• The policies for updating VServer groups vary based on the mode of the Service.
	  ○ If `spec.externalTrafficPolicy = Cluster` is specified for a Service, CCM adds all cluster nodes to the VServer groups of the SLB instance. If node labels are specified in the Service configuration, CCM adds cluster nodes that have the specified labels to the VServer groups of the SLB instance.
	  Notice SLB limits the number of VServer groups to which an ECS instance can be added. If a Service is in Cluster mode, the quota is consumed at a high rate. When the quota is used up, Service reconciliation fails. To fix this issue, set Local mode for a Service.
	  ○ If `spec.externalTrafficPolicy = Local` is specified for a Service, CCM adds only the nodes where the pods that are related to the Service are deployed to the VServer groups of the SLB instance. This can reduce the consumption rate of the resources. Source IP addresses can also be retained in Layer 4 load balancing.
	• CCM does not add master nodes of a cluster to the VServer groups of an SLB instance.
	• Assume that you have run the `kubectl drain` command to remove a node from a cluster, or run the `kubectl cordon` command to mark a node as unschedulable. By default, CCM does not remove such a node from VServer groups of an SLB instance. To remove such a node, set annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend` to `on`.
	  Notice By default, CCM versions earlier than V1.9.3.164-g2105d2e-aliyun remove drained nodes or unschedulable nodes from the backend of the SLB instance.	

Considerations for reusing an existing SLB instance

- **Before you reuse an existing SLB instance, check whether the instance meets the following requirements:**
  - The SLB instance that you want to reuse is created in the SLB console. You cannot reuse an SLB instance that is created by CCM.
  - If you want to reuse an internal-facing SLB instance, the SLB instance and the cluster must be deployed in the same virtual private cloud (VPC).
- CCM configures SLB instances only for `LoadBalancer` type Services.

  Notice If you change `Type=LoadBalancer` to `Type! =LoadBalancer` for a Service, CCM deletes the configuration of the SLB instance from the Service. In this case, you cannot access the Service by using the SLB instance.
- When specific conditions are met, CCM uses a declarative API to automatically update the configuration of an SLB instance based on the Service configuration. If you set annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners:` to `true` for a Service, CCM may overwrite the listener configuration modifications that are made in the SLB console.

> **Notice** If the SLB instance is created or managed by CCM, we recommend that you do not modify the configuration of an SLB instance in the SLB console. Otherwise, CCM may overwrite the configuration and the Service may be unavailable.

Considerations for using CCM to configure an SLB instance

- CCM configures SLB instances only for `LoadBalancer` type Services.

  > **Notice** If you change `Type=LoadBalancer` to `Type! =LoadBalancer` for a Service, CCM deletes the SLB instance that CCM previously created for the Service.

- When specific conditions are met, CCM uses a declarative API to automatically update the configuration of an SLB instance based on the Service configuration. CCM may overwrite the listener configuration modifications that are made in the SLB console.

  > **Notice** If the SLB instance is created or managed by CCM, we recommend that you do not modify the configuration of an SLB instance in the SLB console. Otherwise, CCM may overwrite the configuration and the Service may be unavailable.

Resource quotas

VPC

- A node in a cluster is mapped to a route entry in a route table. By default, each route table for a VPC contains a maximum of 48 entries. If the number of nodes in a cluster exceeds 48, Submit a ticket.

  > **Note** In the ticket, describe your request to modify parameter `vpc_quota_route_entrys_num`. After the request is accepted, the maximum number of custom entries that can be created in one route table is increased.

- For more information about VPC resource quotas, see 限制与配额.
  To query the VPC resource quotas, go to the Quota Management page in the VPC console.

SLB

- CCM creates an SLB instance for a `LoadBalancer` type Service. By default, you can retain a maximum of 60 SLB instances under each account. To create more SLB instances, Submit a ticket.

  > **Note** In the ticket, describe your request to modify parameter `slb_quota_instances_num`. After the request is accepted, the maximum number of SLB instances that can be retained under your account is increased.

- CCM adds ECS instances to the VServer groups of an SLB instance based on the Service configuration.
  - By default, an ECS instance can be added to a maximum of 50 VServer groups. To add the ECS instance to more VServer groups, Submit a ticket.

    > **Note** In the ticket, describe your request to modify parameter `slb_quota_backendservers_num`. After the request is accepted, the maximum number of VServer groups to which an ECS instance can be added is increased.

- By default, a maximum of 200 backend servers can be added to an SLB instance. To add more backend servers, submit a ticket.

  > **Note** In the ticket, describe your request to modify parameter `slb_quota_backendservers_num`. After the request is accepted, the maximum number of backend servers that can be added to an SLB instance is increased.

- CCM creates listeners based on the ports that are specified for a Service. By default, a maximum of 50 listeners can be created for an SLB instance. To create more listeners, submit a ticket.

  > **Note** In the ticket, describe your request to modify parameter `slb_quota_listeners_num`. After the request is accepted, the maximum number of listeners that can be created for an SLB instance is increased.

- For more information about SLB resource quotas, see Limits.
  To query the SLB resource quotas, go to the Quota Management page in the SLB console.

6.4.2. Use annotations to configure load balancing

You can add annotations to the YAML file of a Service to configure load balancing. This topic describes how to use annotations to configure load balancing. You can configure the Server Load Balancer (SLB) instance, listeners, and vServer groups.

Usage notes

- The values of annotations are case-sensitive.
- In `annotations`, the keyword `alicloud` was changed to `alibaba-cloud` on September 11, 2019.
  Examples:
  Before the update: `service.beta.kubernetes.io/alicloud-loadbalancer-id`
  After the update: `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id`
  The system still supports annotations that use the `alicloud` keyword. You do not need to change the existing annotations.

SLB

Common annotations used to configure load balancing

- Create an Internet-facing SLB instance

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 namespace: default
spec:
 ports:
 - port: 80
   protocol: TCP
   targetPort: 80
 selector:
   run: nginx
 type: LoadBalancer
```

- Create an internal-facing SLB instance

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type: "intranet"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

- Create an HTTP-based SLB instance

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

- Create an HTTPS-based SLB instance
  You must first create a certificate in the SLB console. Then, you can use the certificate ID and the following template to create a LoadBalancer Service and an HTTPS-based SLB instance.

  > **Note** HTTPS listeners of the SLB instance decrypt HTTPS requests into HTTP requests and forward the requests to pods on the backend servers.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "https:443"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id: "${YOUR_CERT_ID}"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 443
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

- Create an SLB instance of a specified specification
  For more information about specifications of SLB instances, see CreateLoadBalancer. Use the following template to create a LoadBalancer Service and an SLB instance of a specified specification. You can also use the template to update the specification of an existing SLB instance.

  > **Notice** If you modify the specification of the SLB instance in the SLB console, the modification may be restored by the cloud controller manager (CCM). Proceed with caution.

  ```
  apiVersion: v1
  kind: Service
  metadata:
   annotations:
    service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: "slb.s1.small"
   name: nginx
   namespace: default
  spec:
   ports:
   - port: 443
    protocol: TCP
    targetPort: 443
   selector:
    run: nginx
   type: LoadBalancer
  ```

- Use an existing SLB instance

- By default, the CCM does not overwrite listeners of an existing SLB instance. To overwrite listeners of an existing SLB instance, set the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners` to *true*.

  > **Note** The following list explains why CCM does not overwrite listeners of an existing SLB instance:
  >
  > - If you overwrite the listeners of an existing SLB instance that distributes network traffic to a Service, a Service interruption may occur.
  > - CCM supports limited backend configurations and cannot handle complex configurations. If the vServer group requires complex configurations, you can manually create listeners in the SLB console instead of overwriting the existing listeners.
  >
  > In both cases, we recommend that you do not overwrite the listeners of existing SLB instances. However, you can overwrite an existing listener if the port of the listener is no longer in use.

- You cannot add additional tags to an existing SLB instance by using the `annotation: service.beta.kubernetes.io/alibaba-cloud-loadbalancer-additional-resource-tags` annotation.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id: "${YOUR_LOADBALACER_ID}"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 443
  protocol: TCP
  targetPort: 443
 selector:
  run: nginx
 type: LoadBalancer
```

- Use an existing SLB instance and forcibly overwrite listeners of the SLB instance
  Use the following template to create a LoadBalancer Service and forcibly overwrite an existing listener of the SLB instance. If you specify a different port number, the original port number is overwritten.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id: "${YOUR_LOADBALACER_ID}"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners: "true"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 443
  protocol: TCP
  targetPort: 443
 selector:
  run: nginx
 type: LoadBalancer
```

- Specify primary and secondary zones for an SLB instance

- Primary and secondary zones are not supported by SLB instances that are deployed in some regions, such as the Indonesia (Jakarta) region.
- After you specify the primary and secondary zones for an SLB instance, the zones cannot be changed.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-master-zoneid: "ap-southeast-5a"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-slave-zoneid: "ap-southeast-5a"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

- Create a pay-by-bandwidth SLB instance
  - Only Internet-facing SLB instances support the pay-by-bandwidth billing method.
    For more information about limits on billing methods for Internet-facing SLB instances, see ModifyLoadBalancerInstanceSpec.
  - The annotations in the following template are required.
    Modify the annotation values based on your business requirements.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-charge-type: "paybybandwidth"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-bandwidth: "2"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 443
  protocol: TCP
  targetPort: 443
 selector:
  run: nginx
 type: LoadBalancer
```

> **Note** The annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-bandwidth specifies the maximum bandwidth value.

- Create an SLB instance that has the health check feature enabled

- Enable TCP health checks
  - By default, the health check feature is enabled for TCP listeners and cannot be disabled. The annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-flag annotation` is invalid.
  - To enable TCP health checks, all annotations that are specified in the following example are required.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-type: "tcp"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-timeout: "8"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-healthy-threshold: "4"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-unhealthy-threshold: "4"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval: "3"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

- Enable HTTP health checks
  To enable HTTP health checks, all annotations that are specified in the following example are required.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-flag: "on"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-type: "http"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-uri: "/test/index.html"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-healthy-threshold: "4"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-unhealthy-threshold: "4"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-timeout: "10"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval: "3"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

- Set the routing algorithm for an SLB instance
  - rr: Requests are distributed to backend servers in sequence. This is the default routing algorithm.

- wrr: Backend servers that have higher weights receive more requests than those that have lower weights.
- wlc: Requests are distributed based on the weight and load of each backend server. The load refers to the number of connections to a backend server. If two backend servers have the same weight, the backend server that has fewer connections is expected to receive more requests.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-scheduler: "wlc"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 443
  protocol: TCP
  targetPort: 443
 selector:
  run: nginx
 type: LoadBalancer
```

- Specify a vSwitch for an SLB instance
  - Obtain the ID of a specific vSwitch in the Virtual Private Cloud (VPC) console. Then, use the annotations in the following template to specify the vSwitch for an SLB instance.
  - The specified vSwitch and the cluster must be deployed in the same VPC.
  - All annotations specified in the following template are required.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type: "intranet"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-vswitch-id: "${YOUR_VSWITCH_ID}"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 443
  protocol: TCP
  targetPort: 443
 selector:
  run: nginx
 type: LoadBalancer
```

- Add additional tags to an SLB instance
  Separate multiple tags with commas (,). Example: `"k1=v1,k2=v2"`.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-additional-resource-tags: "Key1=Value1,Key2=Value2
"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

- Create an IPv6 SLB instance
  - Use the following template to create a LoadBalancer Service and an IPv6 SLB instance. The kube-proxy mode must be set to IPVS.
  - The assigned IPv6 address can be used only in an IPv6 network.
  - You cannot change the IP address type after you create the IPv6 SLB instance.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-ip-version: "ipv6"
 name: nginx
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

- Enable deletion protection for an SLB instance
  By default, deletion protection is enabled for SLB instances.

  Notice You may manually enable deletion protection in the SLB console for an SLB instance that is created by a LoadBalancer Service. In this case, you can run the `kubectl delete svc {your-svc-name}` command to delete the SLB instance.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-delete-protection: "on"
 name: nginx
spec:
 externalTrafficPolicy: Local
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

- Enable the configuration read-only mode for an SLB instance
  By default, the configuration read-only mode is enabled for SLB instances.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-modification-protection: "ConsoleProtection"
 name: nginx
spec:
 externalTrafficPolicy: Local
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

- Specify the name of an SLB instance

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-name: "your-svc-name"
 name: nginx
spec:
 externalTrafficPolicy: Local
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

- Specify a resource group to which an SLB instance belongs
  Log on to the Resource Management console to obtain the ID of a specific resource group. Then, use the annotation in the following template to specify the resource group to which the SLB instance belongs.

> **Note** You cannot modify the resource group after the SLB instance is created.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-resource-group-id: "rg-xxxx"
 name: nginx
spec:
 externalTrafficPolicy: Local
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

Listener

Common annotations used to configure listeners

- Set the session persistence period for a TCP-based SLB instance
  - The annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-persistence-timeout` applies to only TCP listeners.
  - If an SLB instance has multiple TCP listeners, the changes apply to all the TCP listeners.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-persistence-timeout: "1800"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 443
  protocol: TCP
  targetPort: 443
 selector:
  run: nginx
 type: LoadBalancer
```

- Enable session persistence for an HTTP/HTTPS-based SLB instance by inserting a cookie
  - The annotations in the following template apply to only HTTP-based and HTTPS-based SLB instances.
  - If an SLB instance has multiple HTTP or HTTPS listeners, the changes apply to all the HTTP or HTTPS listeners.
  - To configure session persistence by inserting a cookie, all annotations in the following example are required.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session: "on"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session-type: "insert"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cookie-timeout: "1800"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "http:80"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

- Enable access control for an SLB instance
  - Create an access control list (ACL) in the SLB console and take note of the ACL ID. Then, use the annotations in the following template to enable access control for an SLB instance.
  - A whitelist allows access from only specified IP addresses. A blacklist denies access from only specific IP addresses.
  - All annotations specified in the following template are required.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-status: "on"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-id: "${YOUR_ACL_ID}"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-type: "white"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 443
  protocol: TCP
  targetPort: 443
 selector:
  run: nginx
 type: LoadBalancer
```

- Configure port forwarding for an SLB instance
  - Port forwarding allows an SLB instance to forward requests from an HTTP port to an HTTPS port.
  - Create a certificate in the SLB console and take note of the certificate ID. Then, use the following annotations to configure port forwarding for an SLB instance.
  - All annotations specified in the following example are required.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "https:443,http:80"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id: "${YOUR_CERT_ID}"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-forward-port: "80:443"
 name: nginx
 namespace: default
spec:
 ports:
 - name: https
  port: 443
  protocol: TCP
  targetPort: 443
 - name: http
  port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

vServer groups

Common annotations used to configure backend servers

- Add worker nodes that have specified labels as backend servers of an SLB instance
  Separate multiple labels with commas (,). Example: `"k1=v1,k2=v2"`. A node must have all specified labels before you can add the node as a backend server.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-backend-label: "failure-domain.beta.kubernetes.io/zone=ap-southeast-5a"
 name: nginx
 namespace: default
spec:
 ports:
 - port: 443
  protocol: TCP
  targetPort: 443
 selector:
  run: nginx
 type: LoadBalancer
```

- Add the nodes where the pods are deployed as the backend servers of an SLB instance
  - By default, `externalTrafficPolicy` is set to Cluster for a Service. In Cluster mode, all nodes in the cluster are added as backend servers of the SLB instance. In Local mode, only nodes where the pods are deployed are added as backend servers of the SLB instance.

- In Local mode, you must set the routing algorithm to weighted round-robin (WRR).

  > **Note**
  > For CCM V1.9.3.164-g2105d2e-aliyun and later, node weights are calculated based on the number of pods that run on each node for Services whose externalTrafficPolicy is set to **Local**. For more information about node weight calculation, see How does CCM calculate node weights in Local mode?.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-scheduler: "wrr"
 name: nginx
 namespace: default
spec:
 externalTrafficPolicy: Local
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

- Remove backend servers in the **Unschedulable** state from an SLB instance
  - You can run the `kubectl cordon` and `kubectl drain` commands to set a node to the **Unschedulable** state. By default, the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend` is set to off. In this case, nodes in the **Unschedulable** state cannot be removed from vServer groups of an SLB instance.
  - To remove backend servers in the **Unschedulable** state from vServer groups of an SLB instance, set the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend` to on.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend: "on"
 name: nginx
spec:
 externalTrafficPolicy: Local
 ports:
 - name: http
  port: 30080
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

- Add pods that are assigned elastic network interfaces (ENIs) as the backend servers of an SLB instance
  When the Terway network plug-in is used, you can use the annotation `service.beta.kubernetes.io/backend-type: "eni"` to add pods that are assigned ENIs as the backend servers of an SLB instance. This improves network forwarding performance.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/backend-type: "eni"
 name: nginx
spec:
 ports:
 - name: http
  port: 30080
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

Note You can also set `eni` in `service.beta.kubernetes.io/backend-type: "eni"` to `ecs`. This allows you to add Elastic Compute Service (ECS) instances as backend severs of an SLB instance.

Common annotations

- **Common annotations that are used to configure load balancing**

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type	string	The type of SLB instance. Valid values: *internet* and *intranet*. ○ *internet*: accesses the Service over the Internet. This is the default value. The **address type** of the SLB instance must be set to **Internet**. ○ *intranet*: accesses the Service over an internal network. The **address type** of the SLB instance must be set to **intranet**.	*internet*	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-charge-type	string	The billing method of an SLB instance. Valid values: *paybytraffic* and *paybybandwidth*.	*paybytraffic*	CCM V1.9.3 and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id	string	The ID of an SLB instance. You can specify an existing SLB instance by using the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id. By default, if you specify an existing SLB instance, the CCM does not overwrite the listeners of the SLB instance. To overwrite the listeners, set the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners to *true*.	None	CCM V1.9.3.81-gca19cd4-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec	string	The specification of an SLB instance. For more information, see CreateLoadBalancer.	None	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-master-zoneid	string	The ID of the zone for the primary backend server.	None	CCM V1.9.3.10-gfb99107-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-slave-zoneid	string	The ID of the zone for the secondary backend server.	None	CCM V1.9.3.10-gfb99107-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners	string	Specifies whether to overwrite the listeners of an existing SLB instance that is specified.	*false*: The listeners of the existing SLB instance are not overwritten.	CCM V1.9.3.81-gca19cd4-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-bandwidth	string	The bandwidth of the SLB instance. This annotation applies to only Internet-facing SLB instances.	*50*	CCM V1.9.3.10-gfb99107-aliyun and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-scheduler	string	The routing algorithm. Valid values: *wrr, wlc, and rr*. ◦ *wrr*: Backend servers that have higher weights receive more requests than the backend servers that have lower weights. ◦ *wlc*: Requests are distributed based on the weight and load of each backend server. The load refers to the number of connections to a backend server. If two backend servers have the same weight, the backend server that has fewer connections is expected to receive more requests. ◦ *rr*: Requests are distributed to backend servers in sequence. This is the default routing algorithm.	*rr*	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-vswitch-id	string	The ID of the vSwitch to which the SLB instance belongs. To set this annotation, set the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type to intranet.	None	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-additional-resource-tags	string	The tags that you want to add to an SLB instance. Separate multiple tags with commas (,). Example: `"k1=v1,k2=v2"`.	None	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-ip-version	string	The IP protocol of an SLB instance. Valid values: ipv4 and ipv6.	ipv4	CCM V1.9.3.220-g24b1885-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-delete-protection	string	Specifies whether to enable deletion protection for an SLB instance. Valid values: on and off.	on	CCM V1.9.3.313-g748f81e-aliyun and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-modification-protection	string	Specifies whether to enable the configuration read-only mode for an SLB instance. Valid values: ConsoleProtection and NonProtection.	ConsoleProtection	CCM V1.9.3.313-g748f81e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-resource-group-id	string	The resource group to which an SLB instance belongs.	None	CCM V1.9.3.313-g748f81e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-name	string	The name of an SLB instance.	None	CCM V1.9.3.313-g748f81e-aliyun and later

- **Common annotations used to configure listeners**

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port	string	The listening port. Separate multiple ports with commas (,). Example: `https:443,http:80`.	None	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-persistence-timeout	string	The session persistence period. Unit: seconds. This annotation applies to only TCP listeners. Valid values: *0 to 3600*. Default value: *0*. By default, session persistence is disabled. For more information, see CreateLoadBalancerTCPListener.	*0*	CCM V1.9.3 and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session	string	Specifies whether to enable session persistence. Valid values: *on and off*. **Note** This annotation applies to only HTTP and HTTPS listeners. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	*off*	CCM V1.9.3 and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session-type	string	The method that is used to process the cookie. Valid values: ○ *insert*: inserts a cookie. ○ *server*: rewrites a cookie. **Note** ○ This annotation applies to only HTTP and HTTPS listeners. ○ If the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session is set to *on*, you must specify this annotation. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	None	CCM V1.9.3 and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cookie-timeout	string	The timeout period of a cookie. Unit: seconds. Valid values: *1 to 86400*. **Note** If service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session is set to *on* and service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session-type is set to *insert*, you must specify this annotation. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	None	CCM V1.9.3 and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cookie	string	The name of a cookie configured on a backend server. The name must be 1 to 200 characters in length, and can contain only ASCII letters and digits. It cannot contain commas (,), semicolons (;), or spaces. It cannot start with a dollar sign ($). **Note** If service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session is set to *on* and service.beta.kubernetes.io/alibaba-cloud-loadbalancer-sticky-session-type is set to *server*, you must specify this annotation. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	None	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id	string	The ID of a certificate for an SLB instance. You must first upload the certificate in the SLB console.	None	CCM V1.9.3.164-g2105d2e-aliyun and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-flag	string	Specifies whether to enable the health check feature. Valid values: *on and off*. ◦ Default value for TCP listeners: on. You cannot modify the value. ◦ Default value for HTTP listeners: off.	Default value: *off*. This annotation is not required for TCP listeners. By default, the health check feature is enabled for TCP listeners. This feature cannot be disabled.	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-type	string	The type of health check. Valid values: *tcp and http*. For more information, see CreateLoadBalancerTCPListener.	*tcp*	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-uri	string	The URI that is used for health checks. **Note** When the type of health check is TCP, this annotation is not required. For more information, see CreateLoadBalancerTCPListener.	None	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-port	string	The port t that is used for health checks. Valid values: 1 to 65535.	None	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-healthy-threshold	string	The number of consecutive successful health checks that must occur before a backend server is declared healthy. Valid values: 2 to 10. For more information, see CreateLoadBalancerTCPListener.	*3*	CCM V1.9.3 and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-unhealthy-threshold	string	The number of consecutive failed health checks that must occur before a backend server is declared unhealthy. Valid values: *2~10* For more information, see CreateLoadBalancerTCPListener.	*3*	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval	string	The health check interval between two consecutive health checks. Unit: seconds. Valid values: *1 to 50*. For more information, see CreateLoadBalancerTCPListener.	*2*	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-timeout	string	The timeout period to wait for a health check response. Unit: seconds. This annotation applies to TCP health checks. If a backend server does not respond within the specified time period, the health check fails. Valid values: *1 to 300*. **Note** If the value of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-timeout is smaller than that of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval, service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-connect-timeout is invalid. In this case, the value of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval is used as the timeout period. For more information, see CreateLoadBalancerTCPListener.	*5*	CCM V1.9.3 and later

Annotation	Type	Description	Default	Supported CMM version

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-timeout	string	The timeout period to wait for a health check response. Unit: seconds. This annotation applies to HTTP health checks. If a backend server does not respond within the specified time period, the health check fails. Valid values: *1 to 300*. **Note** If the value of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-timeout is smaller than that of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval, service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-timeout is invalid. In this case, the value of service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-interval is used as the timeout period. For more information, see CreateLoadBalancerTCPListener.	*5*	CCM V1.9.3 and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-domain	string	The domain name that is used for health checks. ◦ *$_ip*: the private IP address of a backend server. If you do not set this annotation or set the annotation to $_ip, the SLB instance uses the private IP address of each backend server as the domain name for health checks. ◦ *domain*: The domain name must be 1 to 80 characters in length, and can contain only letters, digits, periods (.),and hyphens (-).	None	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-health-check-httpcode	string	The HTTP status code that specifies that a health check is successful. Separate multiple HTTP status codes with commas (,). Valid values: ◦ *http_2xx* ◦ *http_3xx* ◦ *http_4xx* ◦ *http_5xx* Default value: *http_2xx*.	*http_2xx*	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-status	string	Specifies whether to enable access control for the listeners. Valid values: *on and off*.	*off*	CCM V1.9.3.164-g2105d2e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-id	string	The ID of the ACL to which the listeners are bound. If the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-status is set to on, this annotation is required.	None	CCM V1.9.3.164-g2105d2e-aliyun and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-type	string	The type of ACL. Valid values: *white and black*. ◦ *white*: specifies the ACL as a whitelist. Only requests from the IP addresses or CIDR blocks in the ACL are forwarded. Whitelists apply to scenarios when you want to allow only specified IP addresses to access an application. Proceed with caution when you specify the ACL as a whitelist.If you specify the ACL as a whitelist, errors may occur. After a whitelist is set, the SLB instance forwards only requests from the IP addresses in the whitelist. If you set an empty whitelist, the listeners of the SLB instance forward all requests. ◦ *black*: specifies the ACL as a blacklist. All requests from the IP addresses or CIDR blocks in the ACL are rejected. Blacklists apply to scenarios when you want to block access from specified IP addresses to an application. If you set an empty blacklist, the listeners of the SLB instance forward all requests. If the annotation service.beta.kubernetes.io/alibaba-cloud-loadbalancer-acl-status is set to on, this annotation is required.	None	CCM V1.9.3.164-g2105d2e-aliyun and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-forward-port	string	Redirects HTTP requests to the specified port of an HTTPS listener. Example: `80:443`.	None	CCM V1.9.3.164-g2105d2e-aliyun and later

- **Common annotations that are used to configure vServer groups**

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-backend-label	string	Specifies that worker nodes that have matching labels are added as backend servers of an SLB instance.	None	CCM V1.9.3 and later
externalTrafficPolicy	string	The policy that is used to add nodes as backend servers. Valid values: ○ *Cluster*: adds all nodes as backend servers. ○ *Local*: adds the nodes where the pods are deployed as backend servers.	*Cluster*	CCM V1.9.3 and later
service.beta.kubernetes.io/alibaba-cloud-loadbalancer-remove-unscheduled-backend	string	Removes backend servers in the Unschedulable state from an SLB instance. Valid values: *on and off*.	off	CCM V1.9.3.164-g2105d2e-aliyun and later

Annotation	Type	Description	Default	Supported CMM version
service.beta.kubernetes.io/backend-type	string	The type of backend servers attached to an SLB instance. Valid values: ○ `eni` : adds pods as the backend servers of an SLB instance. This parameter takes effect only in Terway mode. This improves network forwarding performance. ○ `ecs` : adds ECS instances as the backend servers of an SLB instance.	When the Flannel network plug-in is used, the default value is `ecs` . When the Terway network plug-in is used: ○ The default value is `ecs` . This applies to Container Service for Kubernetes (ACK) clusters that were created before August 10, 2020. ○ The default value is `eni` . This applies to ACK clusters that are created after August 10, 2020.	CCM V1.9.3.164-g2105d2e-aliyun and later

6.4.3. Use an existing SLB instance to expose an application

Container Service for Kubernetes (ACK) allows you to use a Server Load Balancer (SLB) instance to expose a Service. To access the Service from outside the cluster, you can use the domain name of the SLB instance or the connection string `<IP:Service port>` . To access the Service from within the cluster, you can use the connection string `<Service name:Service port>` . This topic describes how to use an existing SLB instance to expose an application. An NGINX application is used as an example.

Prerequisites

An SLB instance is created by using the SLB console. The SLB instance is deployed in the region where the cluster is created. For more information about how to create an SLB instance, see Create a CLB instance.

Context

By default, cloud controller manager (CCM) v1.9.3 and later do not automatically configure listeners for existing SLB instances. You can add the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners: "true"` to enable CCM to configure listeners. You can also manually configure listeners for an SLB instance.

You can use the following methods to check the CCM version:

- **Console**: Log on to the ACK console and check the CCM version on the Add-ons page.
  i. Log on to the ACK console.
  ii. In the left-side navigation pane, click **Clusters**.
  iii. On the **Clusters** page, find the cluster that you want to manage, and choose **More > Manage System Components** in the Actions column. On the **Add-ons** page, check the CCM version on the Core Components tab.
- **kubectl**: Run the following command to check the CCM version. This method applies to only dedicated Kubernetes clusters.

```
kubectl get pod -n kube-system -o yaml|grep image:|grep cloud-con|uniq
```

Precautions

- When you use an existing SLB instance to expose an application, take note of the following limits:
  - The SLB instance must be created by using the SLB console. You cannot reuse SLB instances that are automatically created by CCM.
  - To reuse an internal-facing SLB instance in a cluster, the SLB instance and the cluster must be deployed in the same virtual private cloud (VPC).
  - The network type of the SLB instance must be consistent with the connection method of the Service. If the Service supports **public access** (`service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type: "internet"`), the **network type** of the SLB instance must be **Internet-facing**. If the Service supports **internal access** (`service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type: "intranet"`), the **network type** of the SLB instance must be **internal-facing**.
  - The SLB instance must listen on different Service ports if the SLB instance exposes more than one Service.
- CCM configures SLB instances only for `Type=LoadBalancer` Services. CCM does not configure SLB instances for other types of Services.

  Notice When a Service is changed from `Type=LoadBalancer` to another type, CCM deletes the configurations that are added to the SLB instance of the Service. As a result, you can no longer use the SLB instance to access the Service.
- CCM uses a declarative API. CCM automatically updates the configurations of an SLB instance to match the configurations of the exposed Service when specific conditions are met. If you set `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners:` to `true`, the configuration modifications that you add in the SLB console may be overwritten.

  Notice Do not use the SLB console to modify the configurations of the SLB instance that is created and managed by CCM. Otherwise, the modifications may be overwritten and the Service may become inaccessible.

SLB resource quotas

- CCM creates SLB instances for `Type=LoadBalancer` Services. By default, you can have a maximum of 60 SLB instances within each Alibaba Cloud account. To create more than 60 SLB instances, Submit a ticket.

  > **Note** In the ticket, specify that you want to modify the `slb_quota_instances_num` parameter to create more SLB instances.

- CCM automatically creates SLB listeners that use Service ports. By default, each SLB instance supports a maximum of 50 listeners. To create more than 50 listeners for an SLB instance, submit a ticket.

  > **Note** In the ticket, specify that you want to modify the `slb_quota_listeners_num` parameter to create more listeners for each SLB instance.

- CCM automatically adds Elastic Compute Service (ECS) instances to backend server groups of an SLB instance based on the Service configurations.
  - By default, an ECS instance can be added to up to 50 backend server groups. To add an ECS instance to more than 50 server groups, Submit a ticket.

    > **Note** In the ticket, specify that you want to modify the `slb_quota_backendserver_attached_num` parameter to add an ECS instance to more server groups.

  - By default, you can add up to 200 backend servers to an SLB instance. To add more backend servers to an SLB instance, submit a ticket.

    > **Note** In the ticket, specify that you want to modify the `slb_quota_backendservers_num` parameter to add more backend servers to an SLB instance.

  For more information about SLB resource quotas, see Limits. To query SLB resource quotas, go to the Quota Management page in the SLB console.

Step 1: Deploy a sample application

The following section describes how to use the **kubectl** command-line tool to deploy an application. For more information about how to deploy an application by using the ACK console, see Create a stateless application by using a Deployment.

1. Use the following YAML template to create a *my-nginx.yaml* file:

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
 name: my-nginx   #The name of the sample application.
 labels:
  app: nginx
spec:
 replicas: 3     #The number of replicas.
 selector:
  matchLabels:
   app: nginx    #You must specify the same value in the selector of the Service that is used to expose the appl
ication.
 template:
  metadata:
   labels:
    app: nginx
  spec:
  #  nodeSelector:
  #   env: test-team
   containers:
   - name: nginx
    image: registry.aliyuncs.com/acs/netdia:latest    #Replace the value with the image address. Format: <ima
ge_name:tags>.
    ports:
    - containerPort: 80                  #The port must be exposed in the Service.
```

2. Run the following command to deploy the my-nginx application:

```
kubectl apply -f my-nginx.yaml
```

3. Run the following command to check the state of the application:

```
kubectl get deployment my-nginx
```

Sample response:

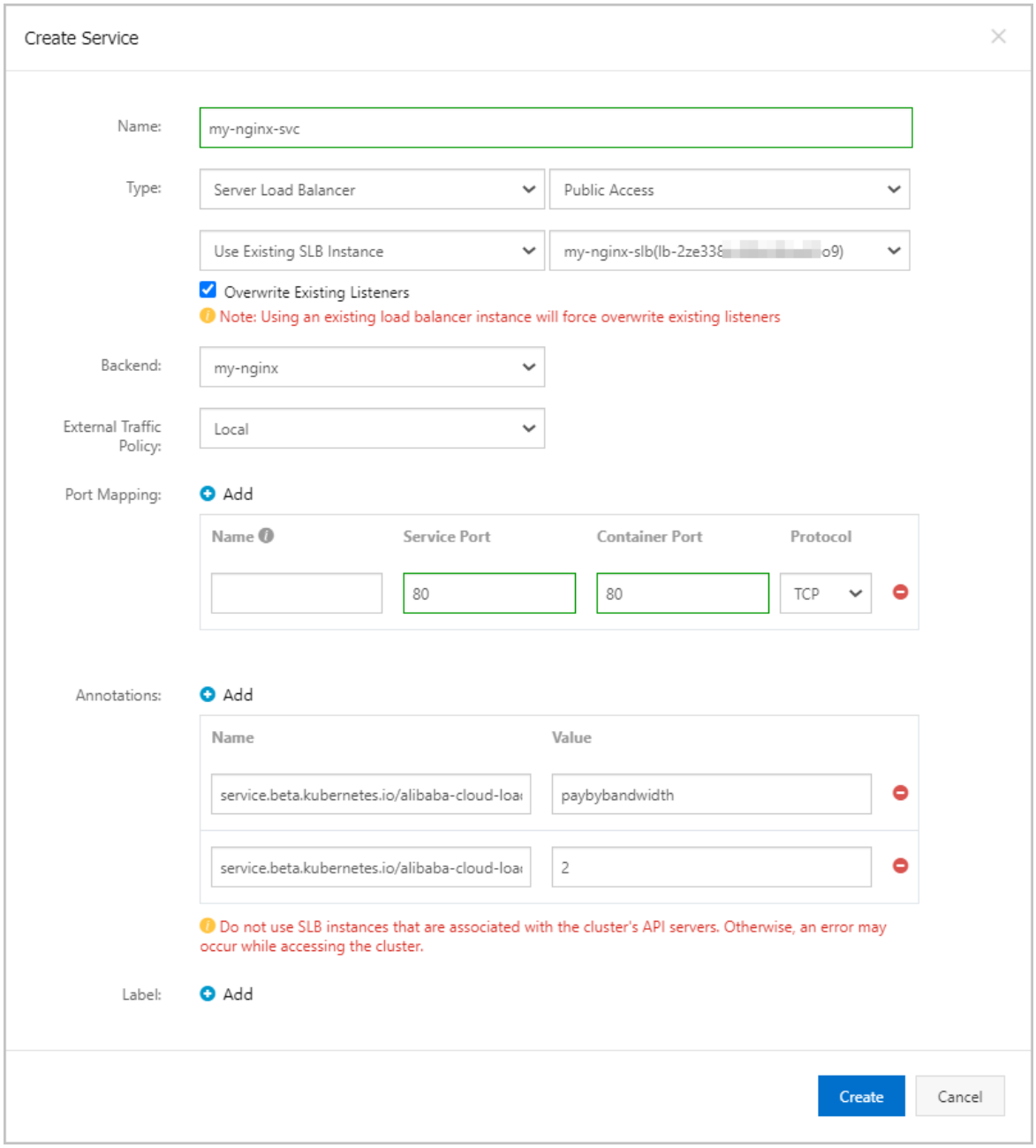
```
NAME    READY  UP-TO-DATE  AVAILABLE  AGE
my-nginx  3/3   3       3       50s
```

Step 2: Use an existing SLB instance to expose the application

You can use the ACK console or **kubectl** to create a LoadBalancer Service. After the Service is created, you can use the Service to expose the application.

Use the ACK console

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Network > Services**.
5. On the **Services** page, click **Create** in the upper-right corner of the page.
6. In the **Create Service** dialog box, set the required parameters.

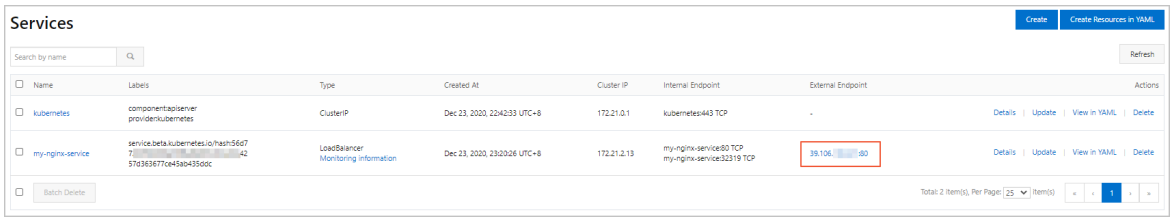


Parameter	Description
Name	Enter a name for the Service. my-nginx-svc is used in this example.

Parameter	Description
Type	Select the type of the Service. This parameter determines how the Service is accessed. ○ Choose **Server Load Balancer** -> **Public Access** -> **Use Existing SLB Instance** and select an SLB instance from the drop-down list. ○ **Overwrite Existing Listeners**: Specify whether to overwrite the listeners of the selected SLB instance. If you select this check box but the SLB instance does not have listeners, the system automatically creates listeners for the SLB instance. In this example, the SLB instance is newly created and therefore you must select this check box to create listeners for the SLB instance. **Note** ■ If the listeners of the SLB instance are associated with applications, service interruptions may occur after the configurations of the listeners are overwritten. ■ CCM supports limited backend configurations and cannot handle complex configurations. If you require complex backend configurations, you can manually modify the listeners in the SLB console without overwriting the existing configurations of the listeners. In both cases, we recommend that you do not overwrite the configurations of the listeners. However, you can overwrite the configuration of a listener if the port of the listener is no longer used.
Backend	Select the application that you want to associate with the Service. The my-nginx application is selected in this example. If you do not associate the Service with a backend, no Endpoint object is created. You can manually associate the Service with a backend. For more information, see services-without-selectors.
External Traffic Policy	Select a policy to distribute external network traffic. **Local** is selected in this example. ○ **Local**: routes network traffic to only pods on the node where the Service is deployed. ○ **Cluster**: routes network traffic to pods on other nodes in the cluster. **Note** The **External Traffic Policy** parameter is available only if you set Type to **Node Port** or **Server Load Balancer**.
Port Mapping	Specify a Service port and a container port. The Service port corresponds to the `port` field in the YAML file and the container port corresponds to the `targetPort` field in the YAML file. The container port must be the same as the one that is exposed in the backend pod. Both ports are set to 80 in this example.
Annotations	Add one or more annotations to the Service to modify the configuration of the SLB instance. You can select **Custom Annotation** or **Alibaba Cloud Annotation** from the Type drop-down list. In this example, the billing method is set to pay-by-bandwidth and the maximum bandwidth is set to 2 Mbit/s to limit the amount of traffic that flows through the Service. For more information, see Use annotations to configure load balancing. ○ **Type**: Alibaba Cloud Annotation ○ **Name**: In this example, two annotations are created with the following names: `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-charge-type` and `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth`. ○ **Value**: In this example, the values of the annotations are set to paybybandwidth and 2.

Parameter	Description
Label	Add one or more labels to the Service. Labels are used to identify the Service.

7. Click **Create**.
   On the **Services** page, you can view the created Service.



8. Click **39.106.XX.XX:80** in the **External Endpoint** column to access the sample application.

Use kubectl

1. Use the following YAML template to create a *my-nginx-svc.yaml* file.
   - Replace the value *${YOUR_LB_ID}* of `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id` with the ID of the SLB instance that is created in the SLB console.
   - If you use an existing SLB instance, CCM does not create listeners for the SLB instance or overwrite the listeners of the SLB instance by default. If you want CCM to create new listeners or overwrite existing listeners, you can set `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners` to `true`. In this example, the SLB instance is newly created and therefore you must set this annotation to `true` to create listeners for the SLB instance. For more information, see Use annotations to configure load balancing.
   - To associate the Service with the backend application, set selector to the value of matchLabels in the *my-nginx.yaml* file. The value is `app: nginx` in this example.

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id: ${YOUR_LB_ID}
  service.beta.kubernetes.io/alicloud-loadbalancer-force-override-listeners: 'true'
 labels:
  app: nignx
 name: my-nginx-svc
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

2. Run the following command to create a Service named my-nginx-svc and use the Service to expose the application:

```
kubectl apply -f my-nginx-svc.yaml
```

3. Run the following command to verify that the LoadBalancer Service is created:

```
kubectl get svc my-nginx-svc
```

Sample response:

```
NAME       TYPE       CLUSTER-IP   EXTERNAL-IP    PORT(S)     AGE
my-nginx-svc  LoadBalancer  172.21.XX.XX  39.106.XX.XX   80:30471/TCP  5m
```

4. Run the **curl <YOUR-External-IP>** command to access the sample application. Replace *<YOUR-External-IP>* with the IP address displayed in the `EXTERNAL-IP` column.

```
curl 39.106.XX.XX
```

Sample response:

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Related information

- Use annotations to configure load balancing
- Use an automatically created SLB instance to expose an application

6.4.4. Use an automatically created SLB instance to expose an application

When no Server Load Balancer (SLB) instance is available, you can use cloud controller manager (CCM) to automatically create an SLB instance for a LoadBalancer Service and then use CCM to manage the SLB instance. This topic describes how to use an automatically created SLB instance to expose an application. An NGINX application is used as an example.

Precautions

- CCM configures SLB instances only for `Type=LoadBalancer` Services. CCM does not configure SLB instances for other types of Services.

> **Notice** When a Service is changed from `Type=LoadBalancer` to another type, CCM deletes the configurations that are added to the SLB instance of the Service. As a result, you can no longer use the SLB instance to access the Service.

- CCM uses a declarative API. CCM automatically updates the configurations of an SLB instance to match the configurations of the exposed Service when specific conditions are met. If you modify the configurations of an SLB instance in the SLB console, CCM may overwrite the changes.

  > **Notice** If an SLB instance is created and managed by CCM, we recommend that you do not modify the configurations of the SLB instance in the SLB console. Otherwise, the changes may be overwritten and you cannot use the SLB instance to access the exposed Service.

SLB resource quotas

- CCM creates SLB instances for `Type=LoadBalancer` Services. By default, you can have a maximum of 60 SLB instances within each Alibaba Cloud account. To create more than 60 SLB instances, Submit a ticket.

  > **Note** In the ticket, specify that you want to modify the `slb_quota_instances_num` parameter to create more SLB instances.

- CCM automatically creates SLB listeners that use Service ports. By default, each SLB instance supports a maximum of 50 listeners. To create more than 50 listeners for an SLB instance, submit a ticket.

  > **Note** In the ticket, specify that you want to modify the `slb_quota_listeners_num` parameter to create more listeners for each SLB instance.

- CCM automatically adds Elastic Compute Service (ECS) instances to backend server groups of an SLB instance based on the Service configurations.
  - By default, an ECS instance can be added to up to 50 backend server groups. To add an ECS instance to more than 50 server groups, Submit a ticket.

    > **Note** In the ticket, specify that you want to modify the `slb_quota_backendserver_attached_num` parameter to add an ECS instance to more server groups.

  - By default, you can add up to 200 backend servers to an SLB instance. To add more backend servers to an SLB instance, submit a ticket.

    > **Note** In the ticket, specify that you want to modify the `slb_quota_backendservers_num` parameter to add more backend servers to an SLB instance.

  For more information about SLB resource quotas, see Limits. To query SLB resource quotas, go to the Quota Management page in the SLB console.

Step 1: Deploy a sample application

The following section describes how to use the **kubectl** command-line tool to deploy an application. For more information about how to deploy an application by using the ACK console, see Create a stateless application by using a Deployment.

1. Use the following YAML template to create a *my-nginx.yaml* file:

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
 name: my-nginx   #The name of the sample application.
 labels:
  app: nginx
spec:
 replicas: 3     #The number of replicas.
 selector:
  matchLabels:
   app: nginx    #You must specify the same value in the selector of the Service that is used to expose the appl
ication.
 template:
  metadata:
   labels:
    app: nginx
  spec:
  #  nodeSelector:
  #   env: test-team
   containers:
   - name: nginx
    image: registry.aliyuncs.com/acs/netdia:latest    #Replace the value with the image address. Format: <ima
ge_name:tags>.
    ports:
    - containerPort: 80                  #The port must be exposed in the Service.
```

2. Run the following command to deploy the my-nginx application:

```
kubectl apply -f my-nginx.yaml
```

3. Run the following command to check the state of the application:

```
kubectl get deployment my-nginx
```

Sample response:

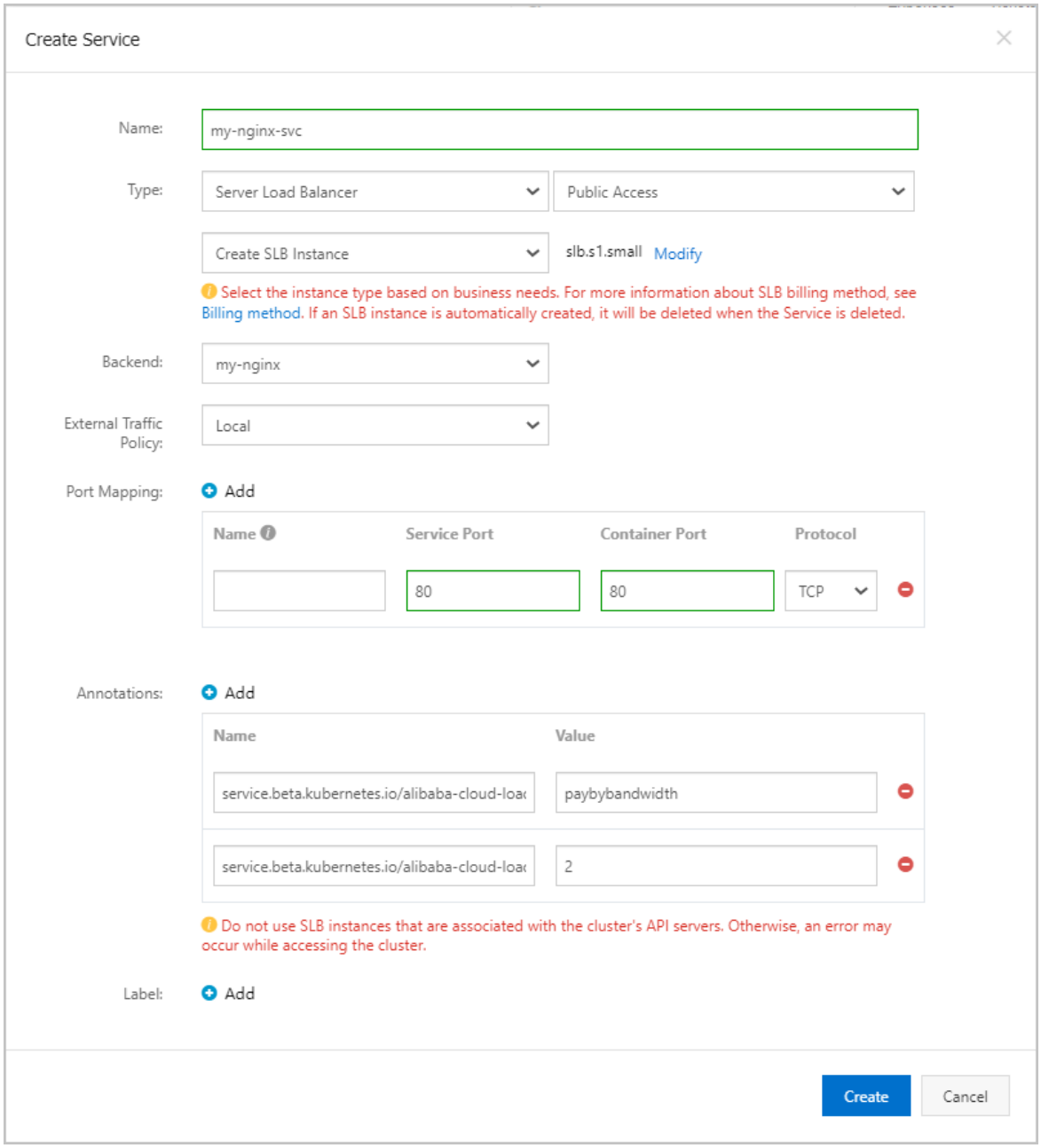
```
NAME    READY  UP-TO-DATE  AVAILABLE  AGE
my-nginx  3/3   3       3       50s
```

Step 2: Use an automatically created SLB instance to expose an application

You can use the Container Service for Kubernetes (ACK) console or **kubectl** to create a LoadBalancer Service. After the Service is created, you use the Service to expose the application.

Use the ACK console

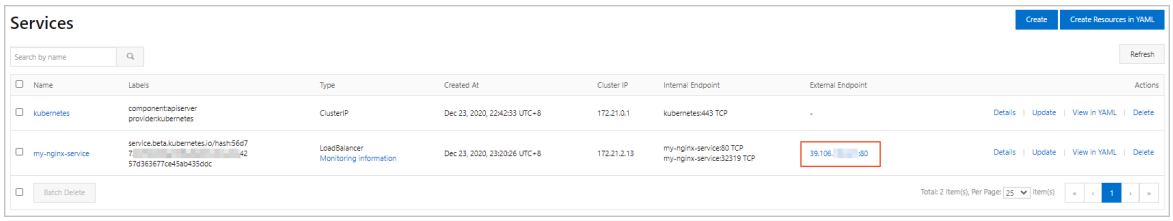
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Network > Services**.
5. On the **Services** page, click **Create** in the upper-right corner of the page.
6. In the **Create Service** dialog box, set the required parameters.



Parameter	Description
Name	Enter a name for the Service. my-nginx-svc is used in this example.
Type	Select a Service type. This parameter determines how the Service is accessed. Choose **Server Load Balancer** -> **Public Access** -> **Create SLB Instance** and click **Modify** to select a specification for the SLB instance. The default specification Small I (slb.s1.small) is used in this example.
Backend	Select the application that you want to associate with the Service. The my-nginx application is selected in this example. If you do not associate the Service with a backend, no Endpoint object is created. You can manually associate the Service with a backend. For more information, see services-without-selectors.

Parameter	Description
External Traffic Policy	Select a policy to distribute external network traffic. **Local** is selected in this example. ○ **Local**: routes network traffic to only pods on the node where the Service is deployed. ○ **Cluster**: routes network traffic to pods on other nodes in the cluster. **Note** The **External Traffic Policy** parameter is available only if you set Type to **Node Port** or **Server Load Balancer**.
Port Mapping	Specify a Service port and a container port. The Service port corresponds to the `port` field in the YAML file and the container port corresponds to the `targetPort` field in the YAML file. The container port must be the same as the one that is exposed in the backend pod. Both ports are set to 80 in this example.
Annotations	Add one or more annotations to the Service to modify the configuration of the SLB instance. You can select **Custom Annotation** or **Alibaba Cloud Annotation** from the Type drop-down list. In this example, the billing method is set to pay-by-bandwidth and the maximum bandwidth is set to 2 Mbit/s to limit the amount of traffic that flows through the Service. For more information, see Use annotations to configure load balancing. ○ **Type**: Alibaba Cloud Annotation ○ **Name**: In this example, two annotations are created with the following names: `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-charge-type` and `service.beta.kubernetes.io/aliclound-loadbalancer-bandwidth`. ○ **Value**: In this example, the values of the annotations are set to paybybandwidth and 2.
Label	Add one or more labels to the Service. Labels are used to identify the Service.

7. Click **Create**.
   On the **Services** page, you can view the created Service.



8. Click **39.106.XX.XX:80** in the **External Endpoint** column to access the sample application.

Use Kubectl

1. Use the following YAML template to create a *my-nginx-svc.yaml* file.

   To associate the Service with the backend application, set selector to the value of matchLabels in the *my-nginx.yaml* file. The value is `app: nginx` in this example.

```
apiVersion: v1
kind: Service
metadata:
 labels:
  app: nignx
 name: my-nginx-svc
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

2. Run the following command to create a Service named my-nginx-svc and use the Service to expose the application:

```
kubectl apply -f my-nginx-svc.yaml
```

3. Run the following command to verify that the LoadBalancer Service is created:

```
kubectl get svc my-nginx-svc
```

Sample response:

```
NAME       TYPE       CLUSTER-IP   EXTERNAL-IP    PORT(S)     AGE
my-nginx-svc  LoadBalancer  172.21.XX.XX  39.106.XX.XX   80:30471/TCP  5m
```

4. Run the **curl <YOUR-External-IP>** command to access the sample application. Replace *<YOUR-External-IP>* with the IP address displayed in the `EXTERNAL-IP` column.

```
curl 39.106.XX.XX
```

Sample response:

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Related information

- Use annotations to configure load balancing
- Use an existing SLB instance to expose an application

6.4.5. Manage Services

Each pod in Kubernetes clusters has its own IP address. However, pods are frequently created and deleted. Therefore, it is not practical to directly expose pods to external access. Services decouple the frontend from the backend, which provides a loosely-coupled microservice architecture. This topic describes how to create, update, and delete Services by using the Container Service for Kubernetes (ACK) console and kubectl.

Manage Services in the ACK console

Create a Service

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Network > Services**.
5. On the Services page, click **Create** in the upper-right corner of the page.
6. In the **Create Service** dialog box, set the parameters.

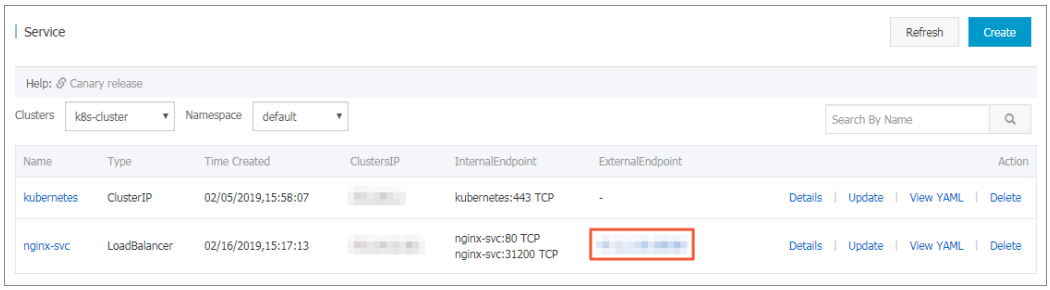
Parameter	Description
Name	Enter a name for the Service.

Parameter	Description
Type	Select the type of the Service. This parameter determines how the Service is accessed. Valid values: ○ Cluster IP: the ClusterIP Service. This type of Service is exposed by using the internal IP address of the cluster. This is the default value. If you select this option, the Service is accessible only from within the cluster. **Note** The **Headless Service** checkbox is available only if you set Type to **Cluster IP**. If you select this check box, you can use a headless Service to interface with other service discovery mechanisms, without being tied to the implementation of service discovery in Kubernetes. ○ Node Port: the NodePort Service. This type of Service is accessed by using the IP address and a static port of each node. The Node Port field specifies the static port. A NodePort Service can be used to route requests to a ClusterIP Service, which is automatically created by the system. You can access a NodePort Service from outside the cluster by sending requests to `<NodeIP>:<NodePort>`. ○ Server Load Balancer: the LoadBalancer Service. This type of Service is accessed by using Server Load Balancer (SLB) instances. If you select this option, you can enable internal or external access to the Service. LoadBalancer Services can be used to route requests to NodePort and ClusterIP Services. ■ Create SLB Instance: You can click **Modify** to change the specification of the SLB instance. ■ Use Existing SLB Instance: You can select an existing SLB instance. **Note** You can create an SLB instance or use an existing SLB instance. You can also associate an SLB instance with more than one Service. However, you must take note of the following limits: ■ If you use an existing SLB instance, the listeners of the SLB instance overwrite those of the Service. ■ If an SLB instance is created along with a Service, you cannot reuse this SLB instance when you create other Services. Otherwise, the SLB instance may be deleted. You can reuse only SLB instances that are manually created in the console or by calling the API. ■ An SLB instance must listen on different Service ports if the SLB instance exposes more than one Service. Otherwise, port conflicts may occur. ■ When you use an SLB instance, the names of listeners and vServer groups are used as unique identifiers in Kubernetes. Do not modify the names of listeners and vServer groups. ■ You cannot use an SLB instance to expose Services across clusters.
Backend	Select the backend application that you want to associate with the Service. If you do not associate the Service with a backend, no Endpoint object is created. You can also manually associate the Service with a backend. For more information, see, see services-without-selectors.

Parameter	Description
External Traffic Policy	Select a policy to distribute external traffic. ○ **Local**: This policy routes external traffic to only pods on the node where the Service is deployed. ○ **Cluster**: This policy routes traffic to pods where the Service is deployed and also to pods on other nodes. **Note** The **External Traffic Policy** parameter is available only if you set Type to **Node Port** or **Server Load Balancer**.
Port Mapping	Specify a Service port and a container port. The Service port corresponds to the `port` field in the YAML file and the container port corresponds to the `targetPort` field in the YAML file. The container port must be the same as the one that is exposed in the backend pod.
Annotations	Add annotations for the Service to modify the configurations of the SLB instance. You can select **Custom Annotation** or **Alibaba Cloud Annotation** from the Type drop-down list. For example, the annotation `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:2` specifies that the maximum bandwidth of the Service is 2 Mbit/s. This limits the amount of traffic that flows through the Service. For more information, see Use annotations to configure load balancing.
Label	Add one or more labels to the Service. The labels are used to identify the Service.

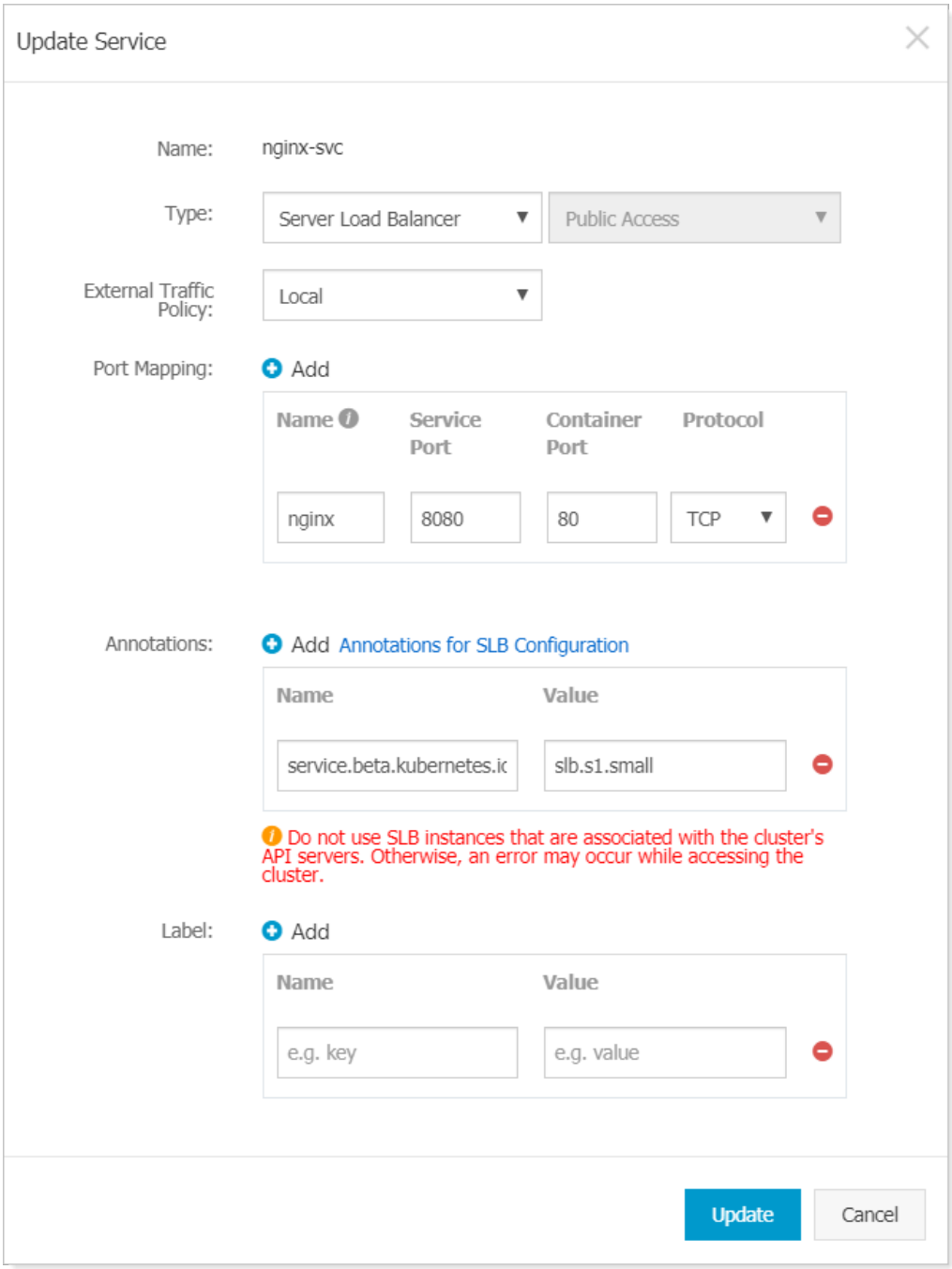
7. Click **Create**.

   On the Services page, you can view the created Service in the Service list.



Update a Service

1. In the left-side navigation pane of the details page, choose **Network > Services**.
2. On the Services page, find the Service that you want to update and click **Update** in the **Actions** column.
3. In the **Update Service** dialog box, set the parameters and click **Update**.

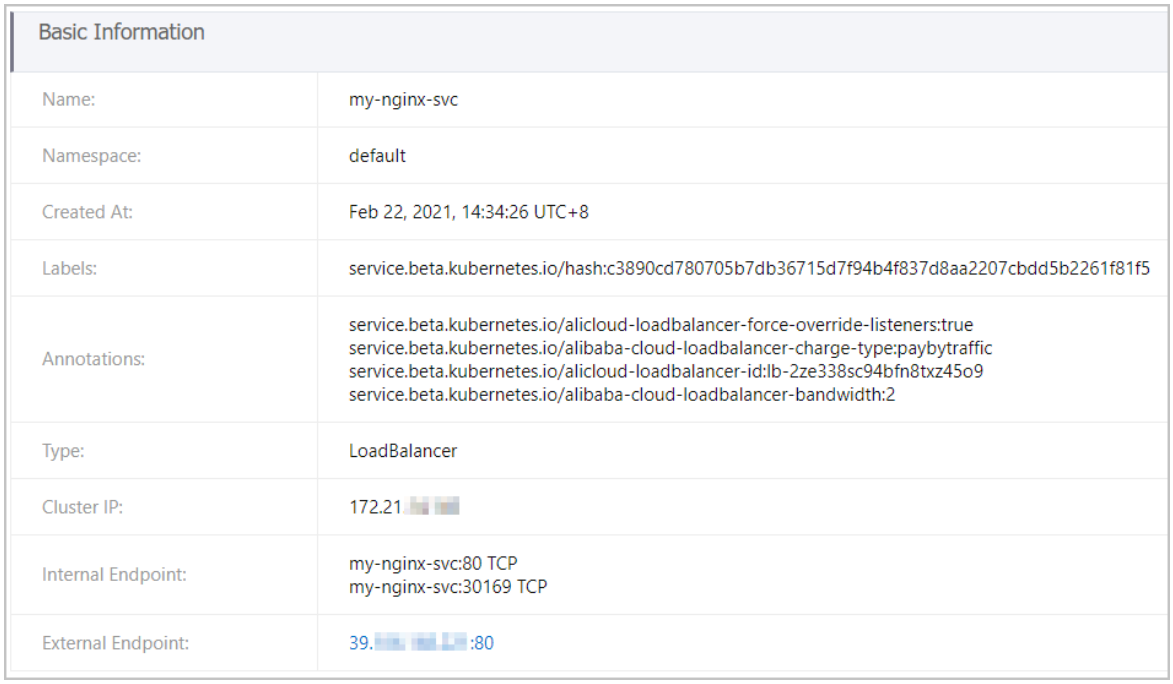


4. In the Service list, find the Service that you updated and click **Details** in the Actions column to view configuration changes.

View a Service

1. In the left-side navigation pane of the details page, choose **Network > Services**.
2. Select a cluster and a namespace, find the Service that you want to view, and then click **Details** in the Actions column.

   You can view information about the Service, such as the name, type, creation time, cluster IP address, and external endpoint. In this example, you can view the external endpoint (the IP address and port) of the Service, as shown in the following figure. To access the NGINX application, click this IP address.



Manage Services by using kubectl

YAML template of a Service

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type: "intranet"
 labels:
  app: nignx
 name: my-nginx-svc
 namespace: default
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: LoadBalancer
```

Field	Description
`kind`	Specifies that the resource object is a Service.
`metadata`	Defines the basic information about the Service, such the name, labels, and namespace.
`metadata.annotations`	ACK provides rich annotations for you to configure load balancing. In this example, the annotation `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type` is set to `intranet`, which specifies that the Service is accessible over the internal network. For more information, see Use annotations to configure load balancing.

Field	Description
spec.selector	Defines the label selector of the Service. The Service exposes the pods with labels that match the label selector.
spec.ports.port	Specifies the Service port that is exposed to the cluster IP address. You can access the Service from within the cluster by sending requests to clusterIP:port.
spec.ports.targetPort	Specifies the port of the backend pod to receive traffic. The traffic that flows through the Service port is forwarded by kube-proxy to the port (specified by targetPort) of the backend pod and then transmitted to the containers.
spec.type	Defines how the Service is accessed. • LoadBalancer: The Service is exposed by using an SLB instance. If you do not associate an existing SLB instance with the Service, the system automatically creates one. By default, the automatically created SLB instance is Internet-facing. You can set service.beta.kubernetes.io/alibaba-cloud-loadbalancer-address-type to intranet to create a Service for internal access and an internal-facing SLB instance for this Service. For more information, see Use an existing SLB instance to expose an application and Use an automatically created SLB instance to expose an application. • ClusterIP: exposes the Service within the cluster. A ClusterIP Service is accessible from within the cluster. • NodePort: maps a node port to the backend Service. You can access the Service from outside the cluster by sending requests to NodeIP:NodePort. • ExternalName: maps the Service to a DNS server.

Create a Service

1. Create a YAML file. For more information, see the preceding YAML template.

   In the following example, a YAML file named *my-nginx-svc.yaml* is created.
2. Connect to the cluster by using kubectl or Cloud Shell. For more information, see Connect to Kubernetes clusters by using kubectl and Use kubectl on Cloud Shell to manage ACK clusters.
3. Run the following command to create a Service:

   ```
   kubectl apply -f my-nginx-svc.yaml
   ```
4. Run the following command to check whether the Service is created:

   ```
   kubectl get svc my-nginx-svc
   ```

   Sample output:

   ```
   NAME        TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)      AGE
   my-nginx-svc  LoadBalancer  172.21.XX.XX  192.168.XX.XX   80:31599/TCP  5m
   ```

Update a Service

- Method 1: Run the following command to update a Service:

  ```
  kubectl edit service my-nginx-svc
  ```
- Method 2: Manually delete a Service, modify the YAML file, and then recreate the Service.

  ```
  kubectl apply -f my-nginx-svc.yaml
  ```

View a Service

Run the following command to view a Service:

```
kubectl get service my-nginx-svc
```

Sample output:

```
NAME       TYPE       CLUSTER-IP   EXTERNAL-IP    PORT(S)     AGE
my-nginx-svc  LoadBalancer  172.21.XX.XX  192.168.XX.XX   80:31599/TCP  5m
```

Delete a Service

Run the following command to delete a Service:

```
kubectl delete service my-nginx-svc
```

6.4.6. FAQ about Services

This topic provides answers to some frequently asked questions about Services of Container Service for Kubernetes (ACK).

FAQ about Server Load Balancer (SLB)

- Why are no events collected during the synchronization between a Service and an SLB instance?
- How do I handle an SLB instance that remains in the Pending state?
- What do I do if the vServer groups of an SLB instance are not updated?
- What do I do if the annotations of a Service do not take effect?
- Why is the configuration of an SLB instance modified?
- Why does the cluster fail to access the IP address of the SLB instance?
- What do I do if I accidentally delete an SLB instance?
- If I delete a Service, is the SLB instance associated with the Service automatically deleted?
- How do I rename an SLB instance when the Cloud Controller Manager (CCM) version is V1.9.3.10 or earlier?
- How does CCM calculate node weights in Local mode?
- How can I use SLB instances in an ACK cluster?

FAQ about CCM upgrades

- How do I troubleshoot failures to upgrade CCM?

FAQ about using existing SLB instances

- Why does the system fail to use an existing SLB instance for more than one Services?
- Why is no listener created when I reuse an existing SLB instance?

Others

- How is session persistence implemented in Kubernetes Services?

Why are no events collected during the synchronization between a Service and an SLB instance?

If no events are collected after you run the `kubectl -n {your-namespace} describe SVC {your-svc-name}` command, verify that your CCM version is **v1.9.3.276-g372aa98-aliyun** or later. If your CCM version is earlier than **v1.9.3.276-g372aa98-aliyun**, no events are collected during the synchronization between a Service and an SLB instance. For more information about how to view and upgrade your CCM version, see Manually upgrade the CCM.

If your CCM version is **v1.9.3.276-g372aa98-aliyun** or later, Submit a ticket.

How do I handle an SLB instance that remains in the Pending state?

1. Run the `kubectl -n {your-namespace} describe svc {your-svc-name}` command to view the events.
2. Troubleshoot the errors that are reported in the events. For more information about how to troubleshoot errors that are reported in the events, see Errors and solutions.

   If no errors are reported in the events, see Why are no events collected during the synchronization between a Service and an SLB instance?.

What do I do if the vServer groups of an SLB instance are not updated?

1. Run the `kubectl -n {your-namespace} describe svc {your-svc-name}` command to view the events.
2. Troubleshoot the errors that are reported in the events. For more information about how to troubleshoot errors that are reported in the events, see Errors and solutions.

   If no errors are reported in the events, see Why are no events collected during the synchronization between a Service and an SLB instance?.

What do I do if the annotations of a Service do not take effect?

1. Perform the following steps to view the errors:
   i. Run the `kubectl -n {your-namespace} describe svc {your-svc-name}` command to view the events.
   ii. Troubleshoot the errors that are reported in the events. For more information about how to troubleshoot errors that are reported in the events, see Errors and solutions.
2. If no errors are reported, you can resolve the issue based on the following scenarios:
   - Make sure that your CCM version meets the requirements of the annotations. For more information about the correlation between annotations and CCM versions, see Common annotations.
   - On the Services page, find the Service that you want to manage and click **View in YAML** in the **Actions** column. In the panel that appears, check whether annotations are configured for the Service. If annotations are not configured for the Service, you must configure annotations for the Service. For more information about how to configure annotations, see Use annotations to configure load balancing.
     For more information about how to view the list of Services, see Manage Services.
   - Verify that the annotations are valid.

Why is the configuration of an SLB instance modified?

When specific conditions are met, CCM calls a declarative API to update the configuration of an SLB instance based on the Service configuration. If you modify the configurations of an SLB instance in the SLB console, CCM may overwrite the configurations. We recommend that you use annotations to configure an SLB instance. For more information about how to configure annotations for an SLB instance, see Use annotations to configure load balancing.

> **Notice** If the SLB instance is created and managed by CCM, we recommend that you do not modify the configuration of the SLB instance in the SLB console. Otherwise, CCM may overwrite the configuration and the Service may be unavailable.

Why does the system fail to use an existing SLB instance for more than one Services?

- Check the version of CCM. If the version is earlier than v1.9.3.105-gfd4e547-aliyun, CCM cannot use existing SLB instances for more than one Services. For more information about how to view and upgrade your CCM version, see Manually upgrade the CCM.
- Check whether the reused SLB instance is created by the cluster. The SLB instance cannot be reused if it is created by the cluster.
- Check whether the SLB instance is used by the API server. The SLB instance cannot be reused if it is used by the API server.
- If the SLB instance is an internal-facing SLB instance, check whether the SLB instance and the cluster are deployed in the same virtual private cloud (VPC). The SLB instance cannot be reused if they are deployed in different VPCs.

Why is no listener created when I reuse an existing SLB instance?

Make sure that you set `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-force-override-listeners` to `true` in the annotation settings. If you do not set the value to true, no listener is automatically created.

> **Note** The following list explains why CCM does not overwrite listeners of an existing SLB instance:
>
> - If you overwrite the listeners of an existing SLB instance that distributes network traffic to a Service, a Service interruption may occur.
> - CCM supports limited backend configurations and cannot handle complex configurations. If the vServer group requires complex configurations, you can manually create listeners in the SLB console instead of overwriting the existing listeners.
>
> In both cases, we recommend that you do not overwrite the listeners of existing SLB instances. However, you can overwrite an existing listener if the port of the listener is no longer in use.

How do I troubleshoot failures to upgrade CCM?

For more information about solutions to CCM upgrade failures, see CCM upgrade failures.

Why does the cluster fail to access the IP address of the SLB instance?

For more information about why does a cluster fail to access the IP address of the SLB instance, see Kubernetes clusters cannot access the IP address of the SLB instance.

If I delete a Service, is the SLB instance associated with the Service automatically deleted?

If the Service has the `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id: {your-slb-id}` annotation which indicates that the SLB instance is reused by the Service, the SLB instance is not deleted after you delete the Service. If the SLB instance is not reused, the SLB instance is deleted when the Service is deleted.

If you change the type of the Service, for example, from LoadBalancer to NodePort, the SLB instance associated with the Service is also deleted.

What do I do if I accidentally delete an SLB instance?

- **Scenario 1: What do I do if I accidentally delete the SLB instance of the API server?**
  The deleted SLB instance cannot be restored. You must create a new SLB instance.
- **Scenario 2: What do I do if I delete the SLB instance of an Ingress?**
  Perform the following steps to recreate the SLB instance:
  i. Log on to the the ACK console.
  ii. In the left-side navigation pane, click **Clusters**.

iii. On the **Clusters** page, find the cluster that you want to manage. Then, click the name of the cluster or click **Details** in the **Actions** column.

iv. In the left-side navigation pane, choose **Network > Services**.

v. On the top of the **Services** page, select **kube-system** from the **Namespace** drop-down list. Then, find **nginx-ingress-lb** in the Services list and click **View in YAML** in the **Actions** column.
If you cannot find **nginx-ingress-lb** in the Services list, use the following template to create a Service named **nginx-ingress-lb**:

```
apiVersion: v1
kind: Service
metadata:
 labels:
  app: nginx-ingress-lb
 name: nginx-ingress-lb
 namespace: kube-system
spec:
 externalTrafficPolicy: Local
 ports:
 - name: http
  port: 80
  protocol: TCP
  targetPort: 80
 - name: https
  port: 443
  protocol: TCP
  targetPort: 443
 selector:
  app: ingress-nginx
 type: LoadBalancer
```

vi. In the **Edit YAML** dialog box, delete the content in the **status** field. Then, click **Update**. This way, CCM creates a new SLB instance.

- **Scenario 3: What do I do if I delete an SLB instance that is configured to handle workloads?**
  - If you no longer need the Service that is associated with the SLB instance, delete the Service.
  - If you want to keep the Service, perform the following steps:
    a. Log on to the the ACK console.
    b. In the left-side navigation pane, click **Clusters**.
    c. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column.
    d. In the left-side navigation pane, choose **Services and Ingresses > Services**.
    e. On the top of the **Services** page, select **All Namespaces** from the **Namespace** drop-down list. Then, find the Service in the Services list and click **View in YAML** in the **Actions** column.
    f. In the **Edit YAML** dialog box, delete the content in the **status** field. Then, click **Update**. This way, CCM creates a new SLB instance.

How do I rename an SLB instance when the CCM version is V1.9.3.10 or earlier?

For CCM versions later than V1.9.3.10, a tag is automatically added to the SLB instances in the cluster. You need only to change the value if you want to rename an SLB instance. For CCM V1.9.3.10 and earlier, you must manually add a specific tag to an SLB instance if you want to rename the SLB instance. Perform the following steps to rename an SLB instance:

> **Note**
> - You can rename an SLB instance by adding a tag to the instance only if the CCM version is V1.9.3.10 or earlier.
> - The Service type is LoadBalancer.

1. Log on to a master node in an ACK cluster. For more information, see Connect to Kubernetes clusters by using kubectl.
2. Run the `# kubectl get svc -n ${namespace} ${service}` command to view the Service type and IP address of the Service.

```
# kubectl get svc -n ${namespace} ${service}
nginx-local   LoadBalancer   172.19.[redacted]   47.111.[redacted]   8900:31598/TCP   33d
```

> **Note** Replace ${namespace} with the cluster namespace and ${service} with the Service name.

3. Run the following command to create the tag that you want to add to the SLB instance:

```
kubectl get svc -n ${namespace} ${service} -o jsonpath="{.metadata.uid}"|awk -F "-" '{print "kubernetes.do.not.delete: "substr("a"$1$2$3$4$5,1,32)}'
```

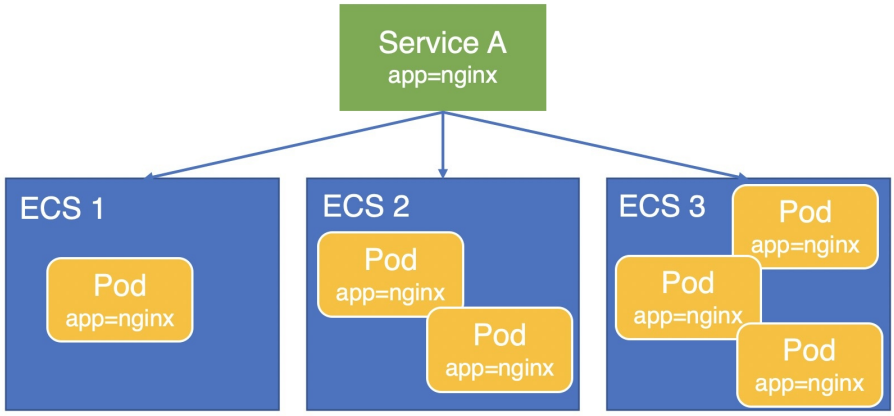
```
# kubectl get svc -n ${namespace} ${service} -o jsonpath="{.metadata.uid}"|awk -F "-" '{print "kub
kubernetes.do.not.delete: a05ff996d0b3a11e999c600163f00d43
```

(1) (2)

4. Log on to the SLB console, select the region where the SLB instance is deployed, and then find the specified SLB instance based on the IP address that is returned in Step 2.
5. Add the tag that is generated in Step 3 to the SLB instance. Callout 1 in the preceding figure is the tag key, and callout 2 is the tag value. For more information, see Add a tag.

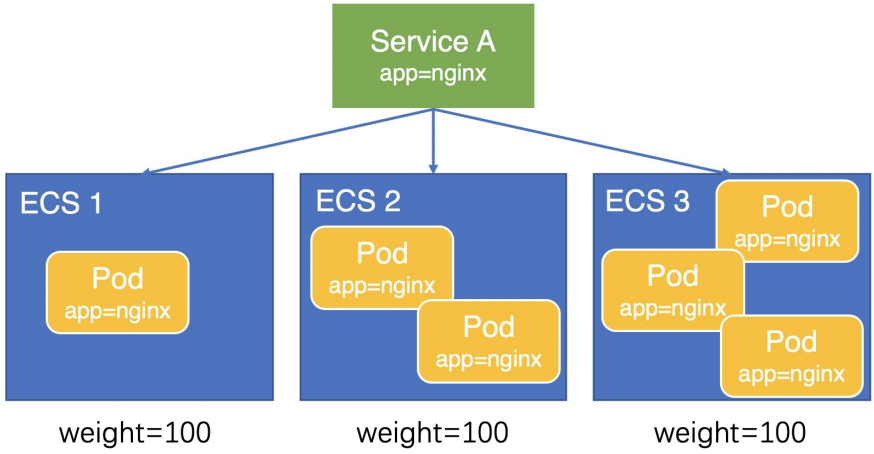
How does CCM calculate node weights in Local mode?

In this example, pods with the app=nginx label are deployed on three ECS instances. In the following figure, when externalTrafficPolicy is set to Local, the pods provide services for external users by using Service A. The following sections describe how node weights are calculated.



- **For CCM versions earlier than V1.9.3.164-g2105d2e-aliyun**

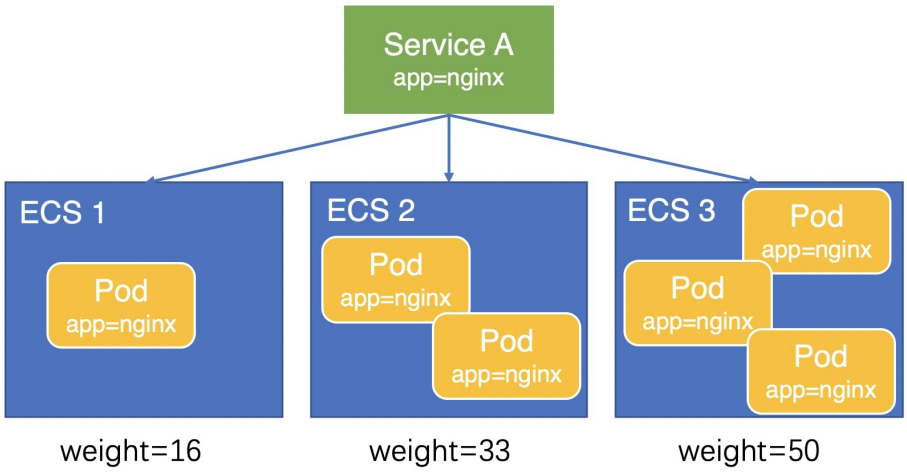
For CCM versions that are earlier than V1.9.3.164-g2105d2e-aliyun, the following figure shows that the weight of each ECS instance in Local mode is 100. This indicates that traffic loads are evenly distributed to the ECS instances. However, the load amounts of the pods are different because the pods are unevenly deployed on the ECS instances. For example, the pod on ECS 1 takes the heaviest load and the pods on ECS 3 take the lightest load.



- **For CCM versions that are later than V1.9.3.164-g2105d2e-aliyun but earlier than V1.9.3.276-g372aa98-aliyun**
  For CCM versions that are later than V1.9.3.164-g2105d2e-aliyun but earlier than V1.9.3.276-g372aa98-aliyun, the node weights are calculated based on the number of pods deployed on each node, as shown in the following figure. The weights of the ECS instances are 16, 33, and 50 based on this calculation. Therefore, traffic loads are distributed to the ECS instances at the ratio of 1:2:3.
  Calculation formula:

$$\text{weight}=\text{node\_pod\_num} \div \text{all\_pod\_num} \times 100$$

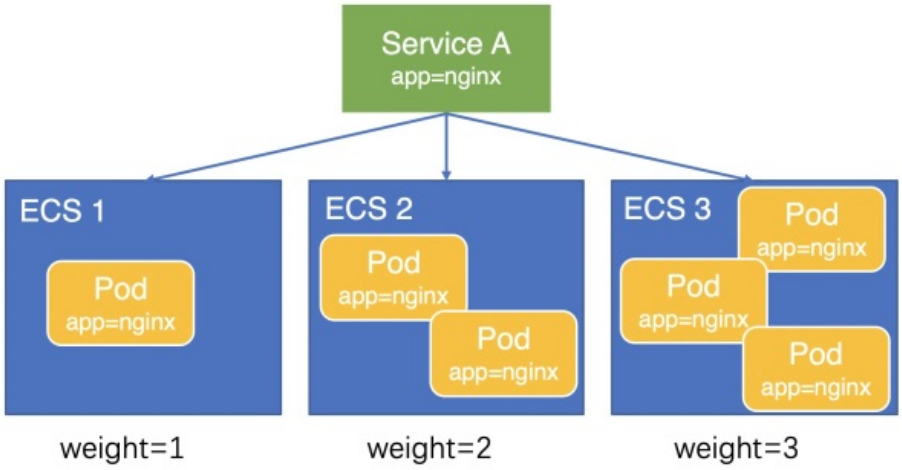
$$\text{node\_weight}=\begin{cases} int(weight) & weight > 1 \\ 1 & weight \le 1 \end{cases}$$



- **For CCM V1.9.3.276-g372aa98-aliyun and later**
  The weights of pods are slightly imbalanced due to the precision of the calculation formula. For CCM V1.9.3.276-g372aa98-aliyun and later, the weight of each node equals the number of pods deployed on the node. In the following figure, the weights of the ECS instances are 1, 2, and 3. Traffic loads are distributed to the ECS instances at the ratio of 1:2:3. This way, the pods have a more balanced load than the pods in the preceding figure.

Calculation formula:

$$node\_weight=\begin{cases} node\_pod\_num & node\_pod\_num < 100 \\ 100 & node\_pod\_num \geq 100 \end{cases}$$



Service errors and solutions

The following table lists solutions to errors occurred in Services.

Error message	Description and solution
The backend server number has reached to the quota limit of this load balancers	The quota of the vServer groups of the SLB instance is insufficient. We recommend that you set externalTrafficPolicy of the SLB instance to Local (`externalTrafficPolicy: Local`) because the quota of vServer groups is quickly consumed in Cluster mode. If you set externalTrafficPolicy of the SLB instance to Cluster, you can specify vServers by adding tags. For more information about how to add tags in annotations to associate with vServers, see Use annotations to configure load balancing.
The backend server number has reached to the quota limit of this load balancer	The number of vServers of the SLB instance has reached the upper limit. If multiple Services share an SLB instance, the number of vServers of each of these Services are accumulated. We recommend that you create a new SLB instance when you create a Service.

Error message	Description and solution
The loadbalancer does not support backend servers of eni type	Shared-resource SLB instances do not support elastic network interfaces (ENIs). If you want to specify an ENI as a backend server, the SLB instance that you create must be a high-performance SLB instance. Add the `annotation: service.beta.kubernetes.io/alibaba-cloud-loadbalancer-spec: "slb.s1.small"` annotation to the Service. **Notice** Make sure that the annotations that you add meet the requirements of the CCM version. For more information about the correlation between annotations and CCM versions, see Common annotations.
alicloud: not able to find loadbalancer named [%s] in openapi, but it's defined in service.loaderbalancer.ingress. this may happen when you removed loadbalancerid annotation	The system fails to associate a Service with the SLB instance. • Check whether the SLB instance exists. If the SLB instance does not exist and the Service is not used, delete the Service. • If the SLB instance exists, perform the following steps: i. Check whether the SLB instance is manually created in the SLB console. If the SLB instance is manually created in the SLB console, reuse the SLB instance by adding the relevant annotation to it. For more information, see Use annotations to configure load balancing. ii. If the SLB instance is automatically created in ACK, check whether the **kubernetes.do.not.delete** label is added to the SLB instance. If the label is not added to the SLB instance, see FAQ about network management.
ORDER.ARREARAGE Message: The account is arrearage.	You have overdue payments.
Status Code: 400 Code: Throttlingxxx	A large number of API operations are called during a specific period of time and API throttling is triggered.

Error message	Description and solution
Status Code: 400 Code: RspoolVipExist Message: there are vips associating with this vServer group.	The listener that is associated with the vServer group cannot be deleted. Solution: 1. Check whether the annotation of the Service contains the ID of the SLB instance, for example, `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-id: {your-slb-id}`. If the annotation of the Service contains the ID of the SLB instance, it indicates that the SLB instance is reused. 2. Log on to the SLB console and delete the listener that uses the Service port. For more information about how to delete listeners for an SLB instance, see Manage forwarding rules for listeners.
Status Code: 400 Code: NetworkConflict	The reused internal-facing SLB instance and the cluster are not deployed in the same VPC. Make sure that your SLB instance and the cluster are in the same VPC.
Status Code: 400 Code: VSwitchAvailableIpNotExist Message: The specified VSwitch has no available ip.	The unused IP addresses in the current vSwitch is insufficient. You can specify another vSwitch in the same VPC by using the `service.beta.kubernetes.io/alibaba-cloud-loadbalancer-vswitch-id: "${YOUR_VSWITCH_ID}"` annotation.
PAY.INSUFFICIENT_BALANCE Message: Your account does not have enough balance.	Your account balance is less than CNY 100.
The specified Port must be between 1 and 65535.	`targetPort` of the STRING type is not supported in ENI mode. Set the `targetPort` field in the Service YAML to an INTEGER value.
Status Code: 400 Code: ShareSlbHaltSales Message: The share instance has been discontinued.	Shared-resource SLB instances are unavailable for purchase. By default, CCM of earlier versions creates shared-resource SLB instances. To resolve this issue, upgrade CCM.

Related information

- Considerations for configuring a LoadBalancer type Service
- Use annotations to configure load balancing
- Cloud Controller Manager

6.5. Ingress management

6.5.1. Ingress overview

This topic introduces Ingresses and describes how Ingress controllers work. It also describes the Ingresses used in Container Service for Kubernetes (ACK) clusters.

What is an Ingress

In a Kubernetes cluster, an Ingress functions as an access point that exposes Services in the cluster. It distributes most of the network traffic that is destined for the Services in the cluster. An Ingress is a Kubernetes resource. It manages external access to the Services in a Kubernetes cluster. You can configure forwarding rules for an Ingress to route network traffic to backend pods of different Services.

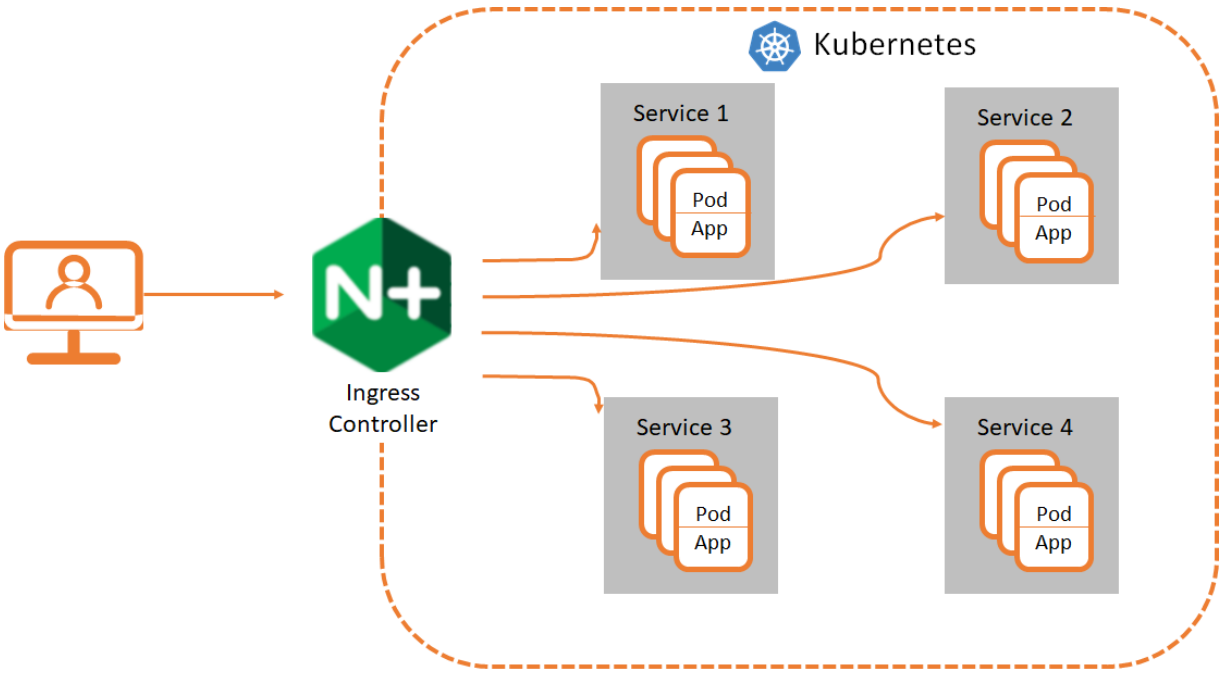
Ingresses can only control HTTP traffic routing by rules. They do not support advanced features such as load balancing algorithms and session affinity. To enable these features, you must configure them on Ingress controllers.

How an Ingress controller works

To ensure that the Ingress resource in a cluster works as expected, you must deploy an Ingress controller in the cluster to parse the Ingress rules. After an Ingress controller receives a request that matches an Ingress rule, the request is routed to the corresponding Service. Then, the Service forwards the request to pods and the pods process the request. In a Kubernetes cluster, Services, Ingresses, and Ingress controllers work in the following process:

- A Service is an abstraction of an application that is deployed on a set of replicated pods.
- An Ingress contains reverse proxy rules. It controls to which Services HTTP or HTTPS requests are routed. For example, an Ingress routes requests to different Services based on the hosts and URLs in the requests.
- An Ingress controller is a reverse proxy program that parses Ingress rules. If changes are made to the Ingress rules, the Ingress controller updates the Ingress rules accordingly. After an Ingress controller receives a request, it redirects the request to a Service based on the Ingress rules.

Ingress controllers acquire Ingress rule changes from the API server and dynamically generate configuration files, such as *nginx.conf*. These configuration files are required by a load balancer, such as NGINX. Then, the Ingress controllers reload the load balancer. For example, the Ingress controllers run the `nginx -s load` command to reload NGINXand then generate new Ingress rules.



Ingress controllers can create Server Load Balancing (SLB) instances for Services of the LoadBalancer type. The SLB instances are used to expose Services in Kubernetes clusters. Ingress rules are then used to control to which Services requests are routed.

Ingresses provided by ACK

ACK provides NGINX Ingress controllers that are optimized based on the open source version. You can choose to install an NGINX Ingress controller when you create an ACK cluster.

Related information

- Ingress高级用法
- Basic operations of an Ingress
- Monitor nginx-ingress and analyze the access log of nginx-ingress
- Use an Ingress controller to mirror network traffic

6.5.2. Basic operations of an Ingress

An Ingress is a Kubernetes resource object that is used to enable external access to Services in a Kubernetes cluster. You can use Ingresses to configure multiple forwarding rules for handling requests to pods in a Kubernetes cluster. This topic describes how to create, view, update, and delete an Ingress in the Container Service for Kubernetes (ACK) console or by using kubectl.

Prerequisites

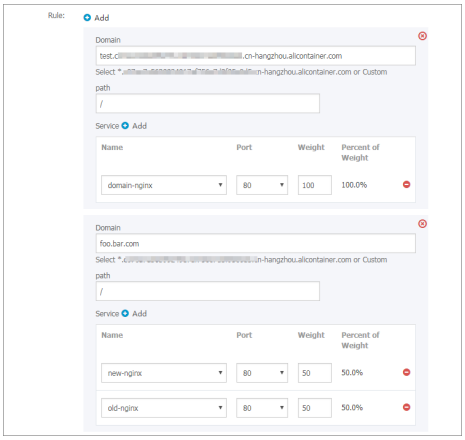
创建Kubernetes托管版集群

How to perform basic operations on an Ingress in the ACK console

Create an Ingress

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Network > Ingresses**.
5. On the **Ingresses** page, click **Create**. In the **Create** dialog box, set the name of the Ingress. In this example, the Ingress is named nginx-ingress.
6. Configure Ingress rules.

   Ingress rules are used to manage external access to Services in the cluster. Ingress rules can be HTTP or HTTPS rules. You can configure the following items in Ingress rules: domain name (virtual hostname), URL path, Service name, port, and weight.
   In this example, a complex rule is added to configure Services for the default domain name and virtual hostname of the cluster. Traffic routing is based on domain names.



   - Create a simple Ingress that uses the default domain name for external access.

- **Domain**: Enter the default domain name of the cluster. In this example, the default domain name is `test.[cluster-id].[region-id].alicontainer.com`.
  In the Create dialog box, the default domain name of the cluster is displayed in the format of `*.[cluster-id].[region-id].alicontainer.com`. You can also obtain the default domain name from the details page of the cluster.
- **Services**: Set the names of the backend Services, and the path and port numbers that are used to access these Services.
  - Path: You can enter the URL for accessing the backend Services. The default path is the root path /. In this example, the default path is used. Each path is associated with a backend Service. Server Load Balancer (SLB) forwards traffic to a backend Service only when inbound requests match the domain name and path.
  - Services: Specify the names of the backend Services and the path and port numbers that are used to access these Services. You can also set weights for these Services. You can configure multiple Services in the same path. The Ingress traffic is split and forwarded to the matched Services.

- Create a simple fanout Ingress that uses multiple domain names. In this example, a virtual hostname is used as the test domain name for external access. Route weights are specified for two backend Services, and canary release settings are configured for one of the Services. In your production environment, you can use a domain name that has obtained an Internet Content Provider (ICP) number for external access.
  - **Domain**: Enter the test domain name. In this example, the test domain name is `foo.bar.com`.
    You must add the following domain name mapping to the hosts file:

    ```
    118.178.XX.XX foo.bar.com    #The IP address of the Ingress.
    ```

  - **Services**: Specify the names of the backend Services, and the path and port numbers that are used to access these Services. You can also set weights for these Services.
    - Path: Enter the URL path of the backend Service. In this example, the default root path / is used.
    - Name: In this example, select the nginx-new and nginx-old Services.
    - Port: In this example, port 80 is opened.
    - Weight: Set a weight for each backend Service. The weight is a percentage value. The default value is 100. In this example, the weight for each backend Service is 50. This means that the two backend Services have the same weight.

7. Configure Transport Layer Security (TLS).

   Select **EnableTLS** to enable TLS and configure a secure Ingress. For more information, see Configure an Ingress.

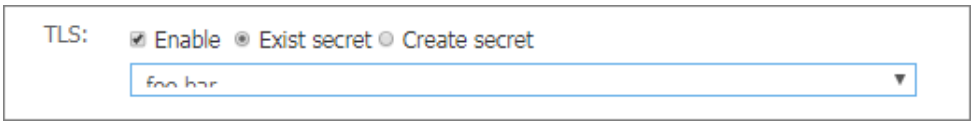
   - You can select an existing Secret.
     a. Log on to a master node. Create a file named *tls.key* and another file named *tls.crt*.

     ```
     openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
     ```

     b. Create a Secret.

     ```
     kubectl create secret tls foo.bar --key tls.key --cert tls.crt
     ```

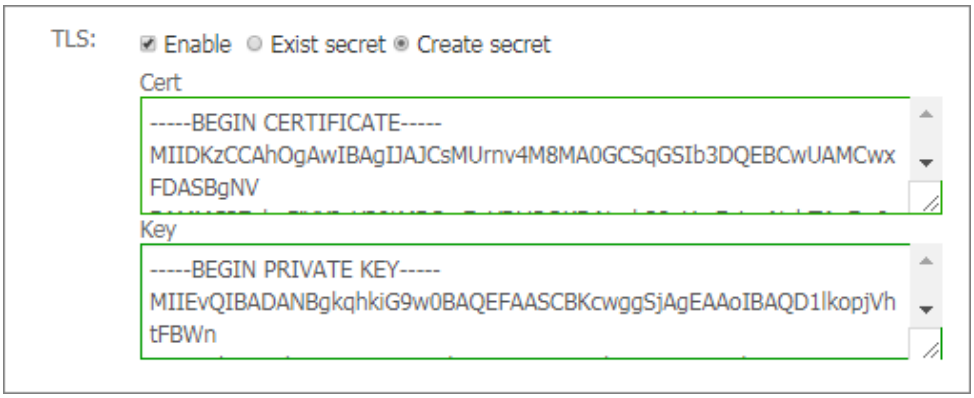
c. Run the `kubectl get secret` command to check whether the Secret is created. Then, you can select the newly created *foo.bar* Secret.



- You can also use the TLS private key and certificate to create a Secret.
  a. Log on to a master node. Create a file named *tls.key* and another file named *tls.crt*.

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
```

  b. Run the `vim tls.key` and `vim tls.crt` commands to obtain the private key and certificate that are generated.
  c. Copy the certificate to the Cert field and the private key to the Key field.

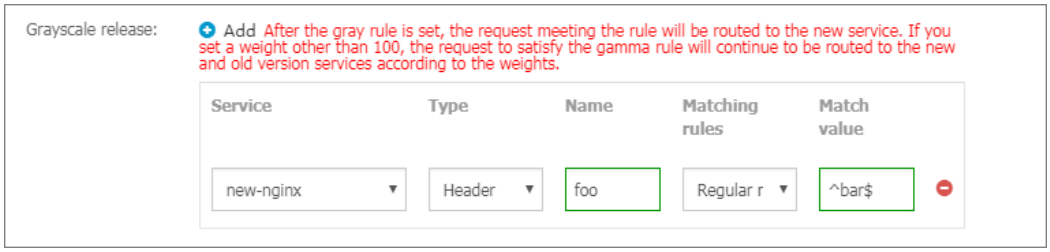


8. Configure canary release settings.

   ACK supports multiple traffic splitting methods. This allows you to select suitable solutions for specific scenarios, such as canary releases and A/B testing.

   i. Traffic splitting based on request headers.
   ii. Traffic splitting based on cookies.
   iii. Traffic splitting based on query parameters.

   After canary release settings are configured, only requests that match the specified rules are routed to the new-nginx Service. If the weight of new-nginx is lower than 100%, requests that match the specified rules are routed to the Service based on the Service weight.
   In this example, the rule is added to specify a request header that matches the regular expression `foo=^bar$`. Only requests with headers that match the regular expression can access new-nginx.



   - **Services**: Select Services as the backend for the Ingress rule.
   - **Type**: Select the type of the matching rule. Valid values: Header, Cookie, and Query.
   - **Name and Match Value**: Specify custom request fields. The name and matching value comprise a

key-value pair.

- **Matching Rule**: Regular expressions and exact matches are supported.

> **Note** You can configure canary release settings for up to two Services.

9. Configure annotations.

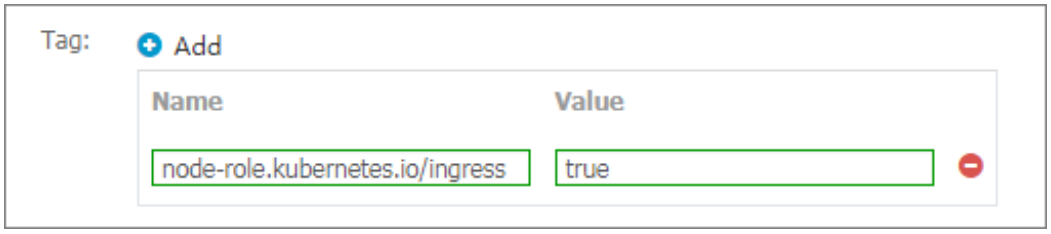
   Click **Add** on the right side of Annotations. In the **Type** drop-down list, you can select the type of annotation based on the following description:

   - **Custom Annotation**: Enter names and values as key-value pairs for the annotation. For more information, see Annotations.
   - **Ingress-NGINX**: Select annotations by names.
     You can add an annotation to redirect inbound traffic. For example, `nginx.ingress.kubernetes.io/rewrite-target: /` specifies that */path* is redirected to the root path */*. The root path can be recognized by the backend Services.

     > **Note** In this example, no path is configured for the backend Services. Therefore, you do not need to configure rewrite annotations. Rewrite annotations allow the Ingress to forward traffic through root paths to the backend Services. This avoids 404 errors that are caused by invalid paths.

10. Add labels.

    Add labels to describe the Ingress.



11. Click **Create**. You are redirected to the Ingresses page.

    After the Ingress is created, you can find the nginx-ingress Ingress on the Ingresses page.

View an Ingress

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Network > Ingresses**.
5. On the **Ingresses** page, find the Ingress that you want to view and click **Details** in the **Actions** column.

   On the details page, you can view detailed information about the Ingress.

Update an Ingress

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Network > Ingresses**.

5. On the **Ingresses** page, find the Ingress that you want to update and click **Details** in the **Actions** column.
6. In the **Update** dialog box, modify the parameters based on your requirements and click **Update**.

Delete an Ingress

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Network > Ingresses**.
5. On the **Ingresses** page, find the Ingress that you want to delete and choose **More > Delete** in the **Actions** column.
6. In the **Note** dialog box, click **Confirm**.

How to perform basic operations on an Ingress by using kubectl

Create an Ingress

1. Create a Deployment and a Service.

   You must create a Service for external access before you can create an Ingress.

i. Create a file named *test-deployment-service.yaml* and copy the following content into the file.

The following YAML template can be used to create a Deployment named test-web1 and a Service named web1-service.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: test-web1
 labels:
  app: test-web1
spec:
 replicas: 1
 selector:
  matchLabels:
   app: test-web1
 template:
  metadata:
   labels:
    app: test-web1
  spec:
   containers:
   - name: test-web1
    imagePullPolicy: IfNotPresent
    image: registry.cn-hangzhou.aliyuncs.com/yilong/ingress-test:web1
    ports:
    - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
 name: web1-service
spec:
 type: ClusterIP
 selector:
  app: test-web1
 ports:
  - port: 8080
   targetPort: 8080
```

ii. Run the following command to create the Deployment and Service:

```
kubectl apply -f test-deployment-service.yaml
```

2. Create an Ingress.

i. Create a file named *test-ingress.yaml* and copy the following content into the file.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: test-ingress
 namespace: default
spec:
 rules:
 - host: test-ingress.com
  http:
   paths:
   - path: /foo
    backend:
     serviceName: web1-service
     servicePort: 8080
   - path: /bar
    backend:
     serviceName: web1-service
     servicePort: 8080
```

- `name`: the name of the Ingress. In this example, the name is set to test-ingress.
- `host`: the domain name for external access to the backend Service.
- `path`: the URL path for external access. Server Load Balancer (SLB) forwards traffic to the `backend` Service only when inbound requests match the `host` and `path` settings.
- `backend`: you need to set a Service and Service port.
  - Service name: the name of the `backend` Service of the Ingress.
  - Service port: the open port of the Service.

ii. Run the following command to create the Ingress:

```
kubectl apply -f test-ingress.yaml
```

View Ingresses

Run the following command to view Ingresses:

```
kubectl get ingress
```

Update an Ingress

Run the following command to update an Ingress:

```
kubectl edit ingress <Ingress name>
```

Delete an Ingress

Run the following command to delete an Ingress:

```
kubectl delete ingress <Ingress name>
```

6.5.3. Use Ingresses to implement canary releases

When you upgrade your application versions, you can implement rolling updates, phased releases, blue-green releases, and canary releases. This topic describes how to implement canary releases for applications in a Container Service for Kubernetes (ACK) cluster by using Ingress controllers.

Prerequisites

创建Kubernetes托管版集群

Context

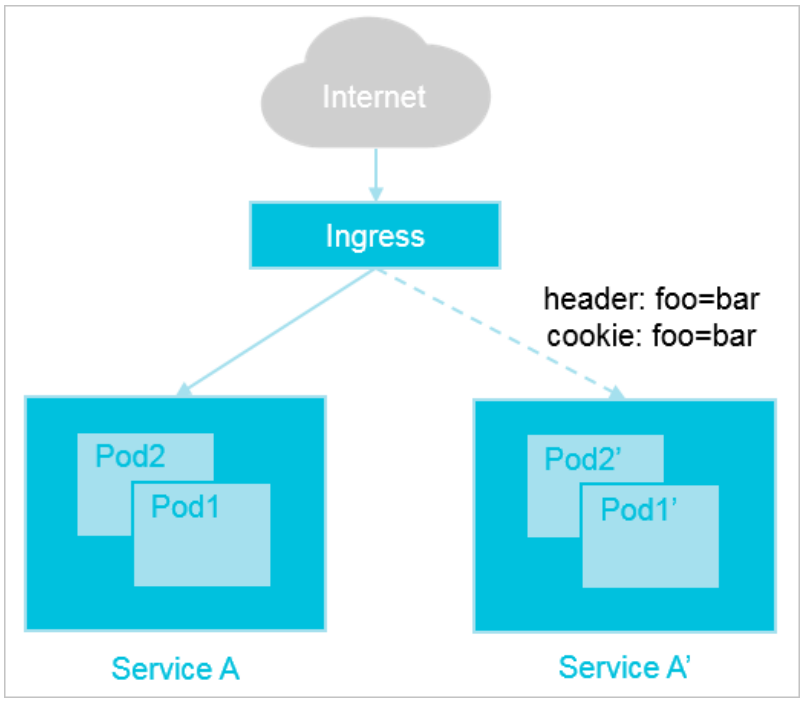
You can implement a canary release or a blue-green release to publish two identical production environments for an earlier application version and a new application version. In this case, when users send requests, ACK routes some requests to the new version based on specific rules. If the new version runs as normal for a period of time, you can switch all traffic from the earlier version to the new version.

A/B testing is a way of implementing canary releases. In A/B testing, some users use the earlier version, and requests from the other users are forwarded to the new version. If the new version runs as normal for a period of time, you can gradually switch all traffic to the new version.

Scenarios

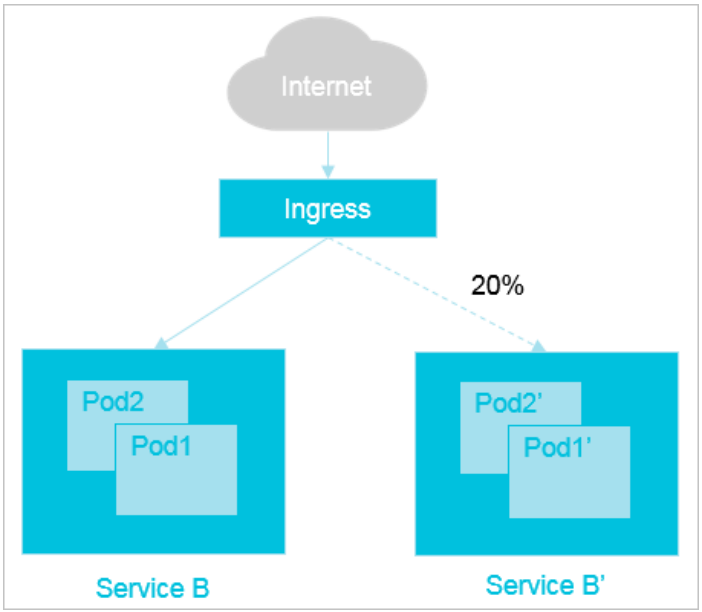
Traffic splitting based on requests

Assume that you have already created Service A for your production environment to provide Layer 7 access for users. When new features are available, you need to create Service A' for the new application version. If you want to keep Service A for external access, you can forward requests whose values of the foo parameters in the request headers match `bar` or whose values of the foo parameters in the cookie match `bar` to Service A'. If the new version stably runs for a period of time, you can switch all traffic from Service A to Service A'. Then, you can delete Service A.



Traffic splitting based on Service weights

Assume that you have already created Service B in your production environment to provide Layer 7 access for users. When some issues are fixed, you need to create Service B' for the new application version. If you want to keep Service B for external access, you can switch 20% of traffic to Service B'. If the new version stably runs for a period of time, you can switch all traffic from Service B to Service B'. Then, you can delete Service B.



Ingress controllers of ACK provide the following traffic splitting methods to support the preceding requirements of application releases.

- Traffic splitting based on request headers. This method applies to scenarios where canary releases or A/B testing is required.
- Traffic splitting based on cookie. This method applies to scenarios where canary releases or A/B testing is required.
- Traffic splitting based on query parameters. This method applies to scenarios where canary releases or A/B testing is required.
- Traffic splitting based on Service weights. This method applies to scenarios where blue-green releases are required.

Annotations

Ingress controllers use the following annotations to implement canary releases of an application.

- nginx.ingress.kubernetes.io/service-match
  This annotation is used to configure match rules for requests to the new application version.

```
nginx.ingress.kubernetes.io/service-match: |
  <service-name>: <match-rule>
# Parameters
# service-name: the name of a Service. Requests that match the rules specified by match-rule are forwarded to the Service.
# match-rule: the match rules for requests.
#
# Match rules:
# 1. Supported match types
# - header: based on the request header. Regular expressions and exact match rules are supported.
# - cookie: based on the cookie. Regular expressions and exact match rules are supported.
# - query: based on the query parameters. Regular expressions and exact match rules are supported.
#
# 2. Match methods
# - Regular expressions: /{regular expression}/. A regular expression is enclosed within forward slashes (/).
# - Exact match rules:"{exact expression}". An exact match rule is enclosed within quotation marks (").
```

Examples of match rules:

```
# If the value of the foo parameter in the request header matches the regular expression ^bar$, the request is forwarded to the new-nginx Service.
new-nginx: header("foo", /^bar$/)
# If the value of the foo parameter in the request header exactly matches the value bar, the request is forwarded to the new-nginx Service.
new-nginx: header("foo", "bar")
# If the value of the foo parameter in the cookie matches the regular expression ^sticky-.+$, the request is forwarded to the new-nginx Service.
new-nginx: cookie("foo", /^sticky-.+$/)
# If the value of the foo parameter in the query parameters exactly matches the value bar, the request is forwarded to the new-nginx Service.
new-nginx: query("foo", "bar")
```

- nginx.ingress.kubernetes.io/service-weight
  This annotation is used to set the weights of the Services for the earlier and new application versions.

```
nginx.ingress.kubernetes.io/service-weight: |
  <new-svc-name>:<new-svc-weight>, <old-svc-name>:<old-svc-weight>
Parameters:
new-svc-name: the name of the Service for the new application version.
new-svc-weight: the traffic weight of the Service for the new application version.
old-svc-name: the name of the Service for the earlier application version.
old-svc-weight: the traffic weight of the Service for the earlier application version.
```

Example of Service weight configurations:

```
nginx.ingress.kubernetes.io/service-weight: |
  new-nginx: 20, old-nginx: 60
```

Step 1: Create an application

Create an NGINX application and deploy an Ingress controller to enable Layer 7 access to the application by using domain names.

1. Create a Deployment and a Service.

i. Create a file named *nginx.yaml*.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: old-nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   run: old-nginx
 template:
  metadata:
   labels:
    run: old-nginx
  spec:
   containers:
   - image: registry.cn-hangzhou.aliyuncs.com/xianlu/old-nginx
    imagePullPolicy: Always
    name: old-nginx
    ports:
    - containerPort: 80
     protocol: TCP
   restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
 name: old-nginx
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: old-nginx
 sessionAffinity: None
 type: NodePort
```

ii. Run the following command to create the Deployment and Service:

```
kubectl apply -f nginx.yaml
```

2. Create an Ingress.

i. Create a file named *ingress.yaml*.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
spec:
 rules:
 - host: www.example.com
   http:
    paths:
    # Configure a Service for the earlier application version.
    - path: /
     backend:
       serviceName: old-nginx
       servicePort: 80
```

ii. Run the following command to create the Ingress:

```
kubectl apply -f ingress.yaml
```

3. Test access to the Ingress.

i. Run the following command to query the IP address of the Ingress for external access:

```
kubectl get ingress
```

ii. Run the following command to access the Ingress:

```
curl -H "Host: www.example.com" http://<EXTERNAL_IP>
```

The following output is returned:

```
old
```

Step 2: Implement a canary release of the application

Release a new NGINX application version and configure Ingress rules.

1. Create a Deployment and a Service for the new application version.

i. Create a file named *nginx1.yaml*.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: new-nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   run: new-nginx
 template:
  metadata:
   labels:
    run: new-nginx
  spec:
   containers:
   - image: registry.cn-hangzhou.aliyuncs.com/xianlu/new-nginx
    imagePullPolicy: Always
    name: new-nginx
    ports:
    - containerPort: 80
     protocol: TCP
   restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
 name: new-nginx
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: new-nginx
 sessionAffinity: None
 type: NodePort
```

ii. Run the following command to create the Deployment and Service:

```
kubectl apply -f nginx1.yaml
```

2. Configure Ingress rules for the new application version.

   ACK provides the following types of Ingress rules. Select a type of Ingress rule based on your requirements.

   - Configure Ingress rules to forward requests that match the rules to the new application version. In the following example, only requests whose values of the foo parameters in the request headers match the regular expression `bar` are forwarded to the new application version.

a. Modify the Ingress that is created in based on the following content.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
 annotations:
  # Only requests whose values of the foo parameters in the request headers match the regular expre
ssion bar are forwarded to the new-nginx Service.
  nginx.ingress.kubernetes.io/service-match: |
   new-nginx: header("foo", /^bar$/)
spec:
 rules:
 - host: www.example.com
  http:
   paths:
   # Configure a Service for the earlier application version.
   - path: /
    backend:
     serviceName: old-nginx
     servicePort: 80
   # Configure a Service for the new application version.
   - path: /
    backend:
     serviceName: new-nginx
     servicePort: 80
```

b. Test access to the Ingress.

- Run the following command to access the NGINX application:

  ```
  curl -H "Host: www.example.com" http://<EXTERNAL_IP>
  ```

  The following output is returned:

  ```
  old
  ```

- Run the following command to access the NGINX application by using a request whose value of the foo parameter in the request header matches the regular expression `bar`:

  ```
  curl -H "Host: www.example.com" -H "foo: bar" http://<EXTERNAL_IP>
  ```

  The following output is returned:

  ```
  new
  ```

You can run the preceding commands again to test the access. The result is that only requests whose values of the foo parameters in the request headers match the regular expression `bar` are forwarded to the new application version.

- Configure Ingress rules to forward a specific proportion of requests that match the rules to the new application version. In the following example, only 50% of the requests whose values of the foo parameters in the request headers match the regular expression `bar` are forwarded to the new version.

a. Modify the Ingress that is created in based on the following content.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
 annotations:
  # Only requests whose values of the foo parameters in the request headers match the regular expre
ssion bar are forwarded to the new-nginx Service.
  nginx.ingress.kubernetes.io/service-match: |
    new-nginx: header("foo", /^bar$/)
  # Only 50% of the requests that match the preceding rule are forwarded to the new-nginx Service.
  nginx.ingress.kubernetes.io/service-weight: |
    new-nginx: 50, old-nginx: 50
spec:
 rules:
 - host: www.example.com
  http:
   paths:
   # Configure a Service for the earlier application version.
   - path: /
    backend:
     serviceName: old-nginx
     servicePort: 80
   # Configure a Service for the new application version.
   - path: /
    backend:
     serviceName: new-nginx
     servicePort: 80
```

b. Test the access to the Ingress.

- Run the following command to access the NGINX application:

  ```
  curl -H "Host: www.example.com" http://<EXTERNAL_IP>
  ```

  The following output is returned:

  ```
  old
  ```

- Run the following command to access the NGINX application by using a request whose value of the foo parameter in the request header matches the regular expression `bar`:

  ```
  curl -H "Host: www.example.com" -H "foo: bar" http://<EXTERNAL_IP>
  ```

  The following output is returned:

  ```
  new
  ```

You can run the preceding commands again to test the access. The result is that only 50% of the requests whose values of the foo parameters in the request headers match the regular expression `bar` are forwarded to the new application version.

- Configure Ingress rules to forward a specific proportion of requests to the new NGINX application. In the following example, only 50% of requests are forwarded to the new NGINX application.

a. Modify the Ingress that is created in based on the following content.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
 annotations:
   # 50% of requests are forwarded to the new-nginx Service.
   nginx.ingress.kubernetes.io/service-weight: |
     new-nginx: 50, old-nginx: 50
spec:
 rules:
 - host: www.example.com
   http:
    paths:
    # Configure a Service for the earlier application version.
    - path: /
      backend:
       serviceName: old-nginx
       servicePort: 80
    # Configure a Service for the new application version.
    - path: /
      backend:
       serviceName: new-nginx
       servicePort: 80
```

b. Run the following command to access the Ingress:

```
curl -H "Host: www.example.com" http://<EXTERNAL_IP>
```

You can run the preceding command again to test the access. The result is that only 50% of the requests are forwarded to the new NGINX application.

Step 3: Delete the earlier application version and the related Service

If the new NGINX application runs as expected for a period of time, you need to bring the earlier application version offline and provide only the new application version for access.

1. Modify the Ingress that is created in based on the following content.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
spec:
 rules:
 - host: www.example.com
   http:
    paths:
    # Configure a Service for the new application version.
    - path: /
      backend:
       serviceName: new-nginx
       servicePort: 80
```

2. Run the following command to access the Ingress:

```
curl -H "Host: www.example.com" http://<EXTERNAL_IP>
```

The following output is returned:

```
new
```

You can run the preceding command again to test the access. The result is that all the requests are forwarded to the new NGINX application.

3. Delete the Deployment and Service for the earlier NGINX application.
   i. Run the following command to delete the Deployment for the earlier NGINX application:

   ```
   kubectl delete deploy <Deployment name>
   ```

   ii. Run the following command to delete the Service for the earlier NGINX application:

   ```
   kubectl delete svc <Service name>
   ```

6.5.4. Enable Tracing Analysis for Ingresses

You can enable Tracing Analysis for Ingresses in an Container Service for Kubernetes (ACK) cluster. Tracing data is imported to the Tracing Analysis console. You can view traces and the trace topology in the Tracing Analysis console. This topic describes how to enable Tracing Analysis for Ingresses.

Prerequisites

- Activate related services and grant required permissions
- 创建Kubernetes托管版集群

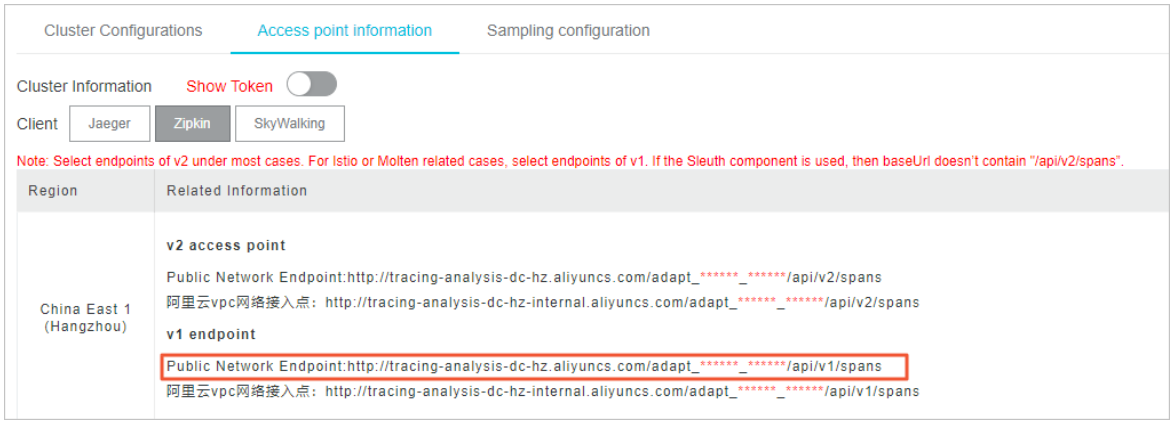
Context

Tracing Analysis provides various features for distributed applications, including trace mapping, request counting, and trace topology. These features allow you to analyze and diagnose the performance bottlenecks of distributed applications, and improve the efficiency of microservice development and diagnostics. You can enable Tracing Analysis for NGINX Ingress controllers that are provided by ACK based on your requirements. After Tracing Analysis is enabled, you can view the tracing data.

Step 1: Obtain an endpoint

Obtain an endpoint. Select a client based on your requirements. In this example, a Zipkin client is used.

1. Log on Tracing Analysis console.
2. In the left-side navigation pane, click **Cluster Configurations**.
3. On the **Cluster Configurations** page, click the **Access point information** tab. Then, turn on **Show Token** for the **Cluster Information** parameter.
4. Set the **Client** parameter to **Zipkin**.
5. In the Related Information column of the table in the lower part, click the copy icon next to the endpoint that you want to use.



Note If you deploy your application in an Alibaba Cloud production environment, select a private endpoint. Otherwise, select a public endpoint.

Step 2: Enable Tracing Analysis for Ingresses

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
5. On the ConfigMap page, enter nginx-configuration into the Name search box and click the search icon. Find the nginx-configuration ConfigMap and click **Edit** in the **Actions** column.
6. Configure Tracing Analysis.

   Edit the ConfigMap based on the client that is used. In this example, a Zipkin client is used.
   In the **Edit** panel, click **Add**. Set **Name** to zipkin-collector-host. Set **Value** to the endpoint that is obtained in Step 1: Obtain an endpoint.
7. Enable Tracing Analysis.

   Click **Add**. Set **Name** to enable-opentracing. Set **Value** to true. Then, click **OK**.

Step 3: View the tracing data

1. Log on Tracing Analysis console.
2. In the left-side navigation pane, click **Applications**.
3. On the **Applications** page, select a region at the top of the page and click the application that you want to manage.
4. In the left-side navigation pane of the details page, click **Interface Calls**. On the page that appears, click the Traces tab. You can view up to 100 traces on the tab. These traces are sorted in descending order of elapsed time. For more information about the tracing data, see View the details of span calls.

6.5.5. Monitor nginx-ingress and analyze the access log of nginx-ingress

Container Service for Kubernetes (ACK) allows you to configure the nginx-ingress component for an ACK cluster. This component provides URLs that can be visited by servers outside the cluster, and supports server load balancing, SSL termination, and name-based virtual hosting. You can also use nginx-ingress to write the log data of HTTP requests to the stdout file. You can enable Log Service for an ACK cluster when you create the cluster. After Log Service is enabled, you can monitor nginx-ingress in real time and view dashboards in Log Service. The dashboards show statistics that are collected from the access log of nginx-ingress.

Prerequisites

1. The alibaba-log-controller component is installed.
   By default, alibaba-log-controller is installed when you create a cluster. If this component is not installed, you can manually install it. For more information, see Collect log files from containers by using Log Service.
2. The alibaba-log-controller component is upgraded.
   Run the `kubectl edit deployment alibaba-log-controller -n kube-system` command to upgrade the component.

Precautions

Take note of the following limits when you deploy the custom resource definition (CRD) for AliyunLogConfig:

- Make sure that the version of alibaba-log-controller is `0.2.0.0-76648ee-aliyun` or later. After you upgrade alibaba-log-controller, if you find that the CRD is already deployed, delete the CRD and redeploy it.
- The CRD for AliyunLogConfig applies only to the default log format that ACK defines for the Ingress controller. If you have changed the log format, you must modify the `processor_regex` settings in the nginx-ingress.yaml file. For more information, see Use CRDs to collect Kubernetes container logs in DaemonSet mode.
- The `IncludeLabel` label specifies the label information retrieved by using the docker inspect command.
- A namespace and a container name in a Kubernetes cluster are separately mapped to the following Docker labels: `io.kubernetes.pod.namespace` and `io.kubernetes.container.name`. For example, the pod that you create belongs to the backend-prod namespace and the container name is worker-server.
  - If you set the `io.kubernetes.pod.namespace : backend-prod` whitelist, log files of all containers in the specified pod are collected.
  - If you set the `io.kubernetes.container.name : worker-server` whitelist, log files of the specified container are collected.
- In a Kubernetes cluster, we recommend that you specify only the `io.kubernetes.pod.namespace` and `io.kubernetes.container.name` labels. In other cases, you can specify the IncludeEnv or ExcludeEnv label. For more information, see Use the console to collect Kubernetes stdout and stderr logs in DaemonSet mode.

Create a CRD to collect the log files of nginx-ingress

You can create CRDs to configure log collection in ACK clusters. You can deploy a CRD for AliyunLogConfig. alibaba-log-controller automatically generates configurations for Log Service to collect log data and update data in relevant dashboards.

Run the following command to deploy the CRD for AliyunLogConfig:

```
cat <<-EOF | kubectl apply -n kube-system -f -
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  # your config name, must be unique in you k8s cluster
  name: k8s-nginx-ingress
spec:
  # logstore name to upload log
  logstore: nginx-ingress
```

```
  # product code, only for k8s nginx ingress
  productCode: k8s-nginx-ingress
  # logtail config detail
  logtailConfig:
   inputType: plugin
   # logtail config name, should be same with [metadata.name]
   configName: k8s-nginx-ingress
   inputDetail:
    plugin:
     inputs:
     - type: service_docker_stdout
       detail:
         IncludeLabel:
           io.kubernetes.container.name: nginx-ingress-controller  #If multiple Ingress controllers are configured, dup
licate log files may be collected. Read the description of IncludeLabel in the Precautions section.
         Stderr: false
         Stdout: true
     processors:
     - type: processor_regex
       detail:
         KeepSource: false
         Keys:
         - client_ip
         - x_forward_for
         - remote_user
         - time
         - method
         - url
         - version
         - status
         - body_bytes_sent
         - http_referer
         - http_user_agent
         - request_length
         - request_time
         - proxy_upstream_name
         - upstream_addr
         - upstream_response_length
         - upstream_response_time
         - upstream_status
         - req_id
         - host
         - proxy_alternative_upstream_name
         NoKeyError: true
         NoMatchError: true
         Regex: ^(\S+)\s-\s\[([^]]+)]\s-\s(\S+)\s\[(\S+)\s\S+\s"(\w+)\s(\S+)\s([^"]+)"\s(\d+)\s(\d+)\s"([^"]*)"\s"([^"]*)"\s(\
S+)\s(\S+)+\s\[([^]]*)]\s(\S+)\s(\S+)\s(\S+)\s(\S+)\s(\S+)\s*(\S*)\s*\[*([^]]*)\]*.*
         SourceKey: content
EOF
```

View the log data of nginx-ingress and dashboards

1. Log on to the Log Service console.
2. In the Projects section, click the name of the project that you specified when you created the ACK cluster. The details page of the project appears. By default, a project that is named in the format of

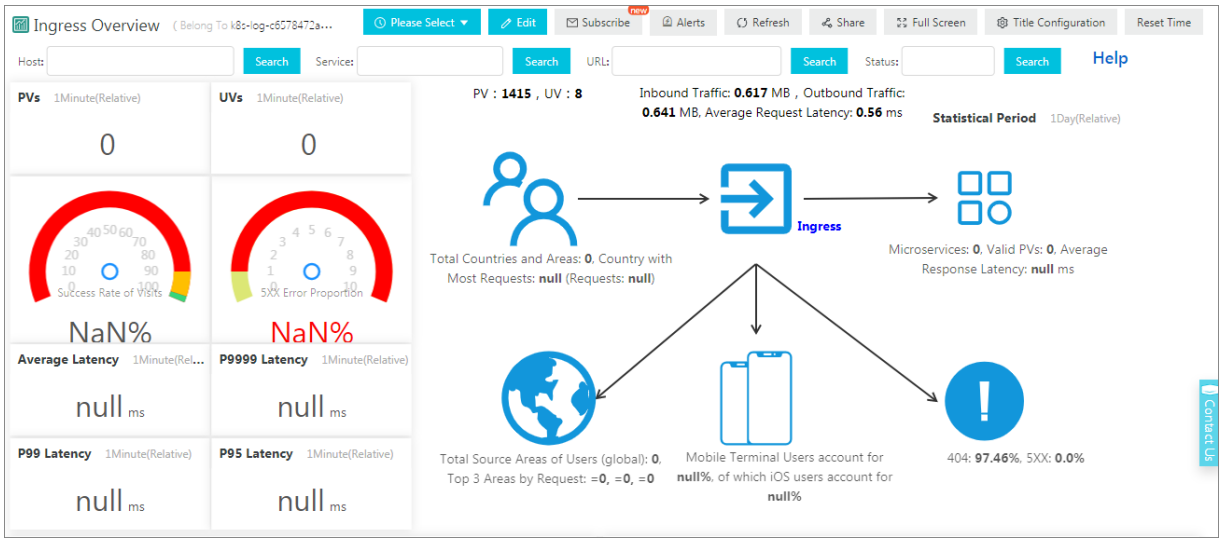
k8s-log-{cluster-id} is created for the ACK cluster.

3. In the left-side navigation pane, choose **nginx-ingress > Visual Dashboards** to view the dashboards of nginx-ingress.

Ingress overview

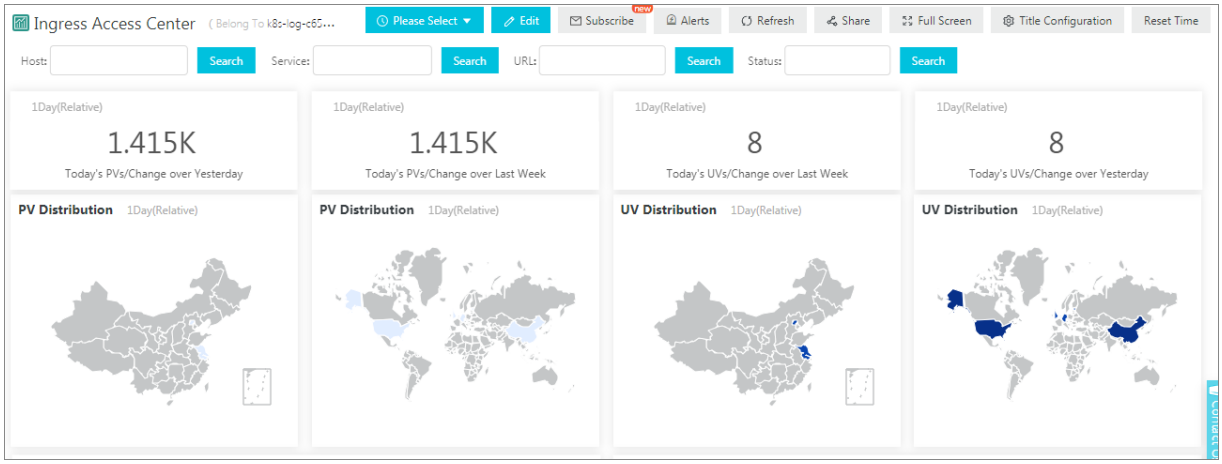
The Ingress overview dashboard displays information about network traffic that flows through nginx-ingress for a website. You can view the following information:

- Website data of the last 24 hours, including the number of page views (PVs), the number of unique visitors (UVs), inbound and outbound traffic, the average latency, the proportion of mobile users, and the proportions of 5xx errors and 404 errors.
- Website data of the last 1 minute, including the number of PVs, the number of UVs, the success rate of requests, the proportion of 5xx errors, the average latency, the P95 latency, and the P99 latency.
- Detailed information about requests within the last 24 hours, including the PV trend (compared with the PV trend within the last 24 hours and the last 7 days), regional distribution of request sources, the top N source areas and cities, the proportion of mobile users, and the proportions of Android users and iOS users.
- Top N URLs within the last 1 hour, including the 10 URLs of the highest PVs, the 10 URLs of the highest latencies, the 10 URLs that return the most 5xx errors, and the 10 URLs that return the most 404 errors.



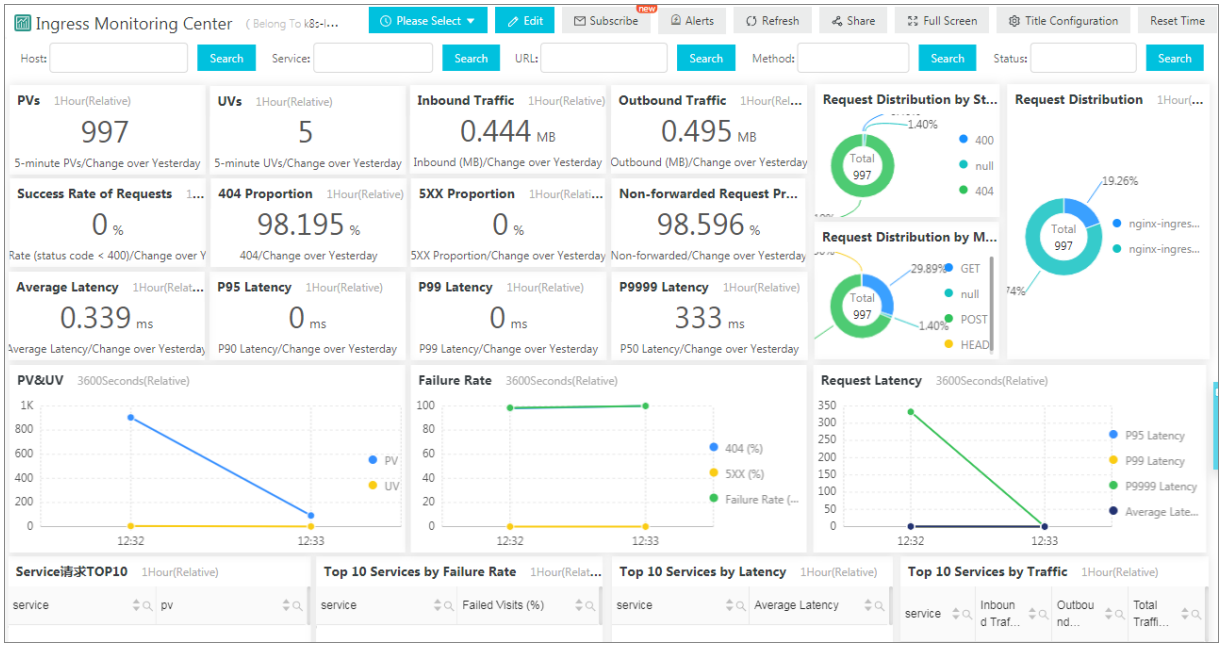
Ingress access center

The Ingress access center dashboard displays up-to-date information about user requests. You can obtain and analyze the following data to help make business decisions: the numbers of UVs and PVs within the last 24 hours, the regional distribution of PVs and UVs, top N areas by request, top N cities by request, top N browsers of the highest PVs, top N source IP addresses of the highest PVs, the proportion of mobile users, and the proportions of Android users and iOS users.



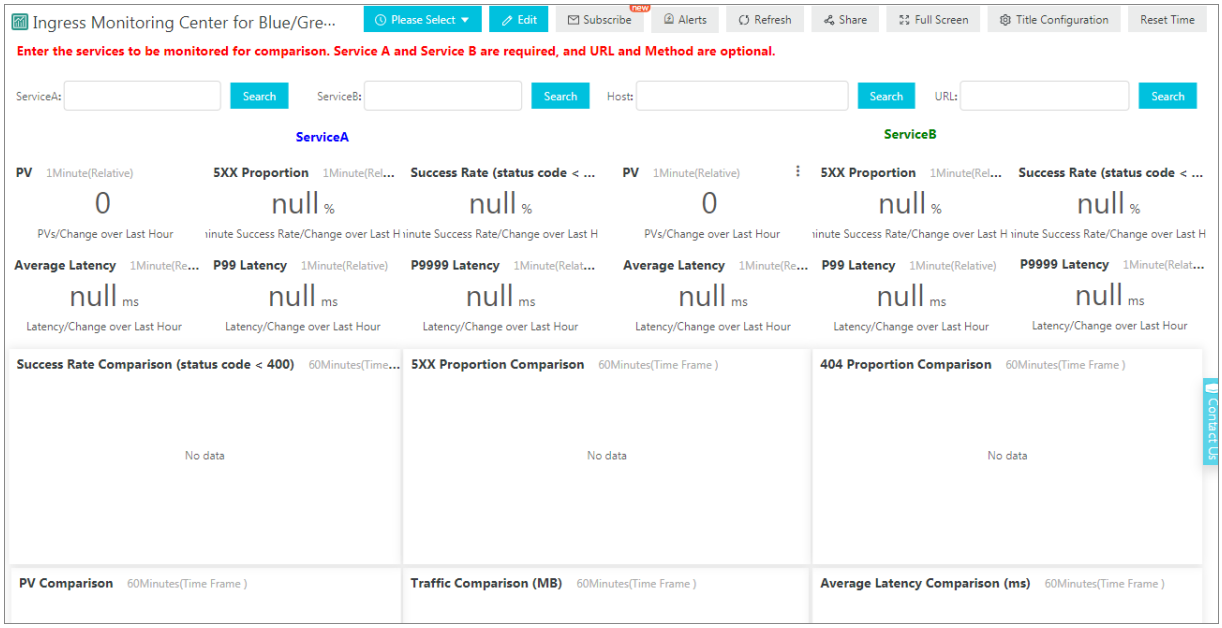
Ingress monitoring center

The Ingress monitoring center dashboard provides real-time monitoring and alerting data of the website. You can view the following data within the last 1 hour: the success rate of requests, the proportions of 5xx errors, 404 errors, and requests that are not forwarded, the average latency, the P95, P99, and P999 latencies, the request distribution by status code, the proportion of PVs on each Ingress, top 10 Services of the highest PVs, top 10 Services of the highest request failure rates, top 10 Services of the highest average latencies, and top 10 Services of the highest data transfer.



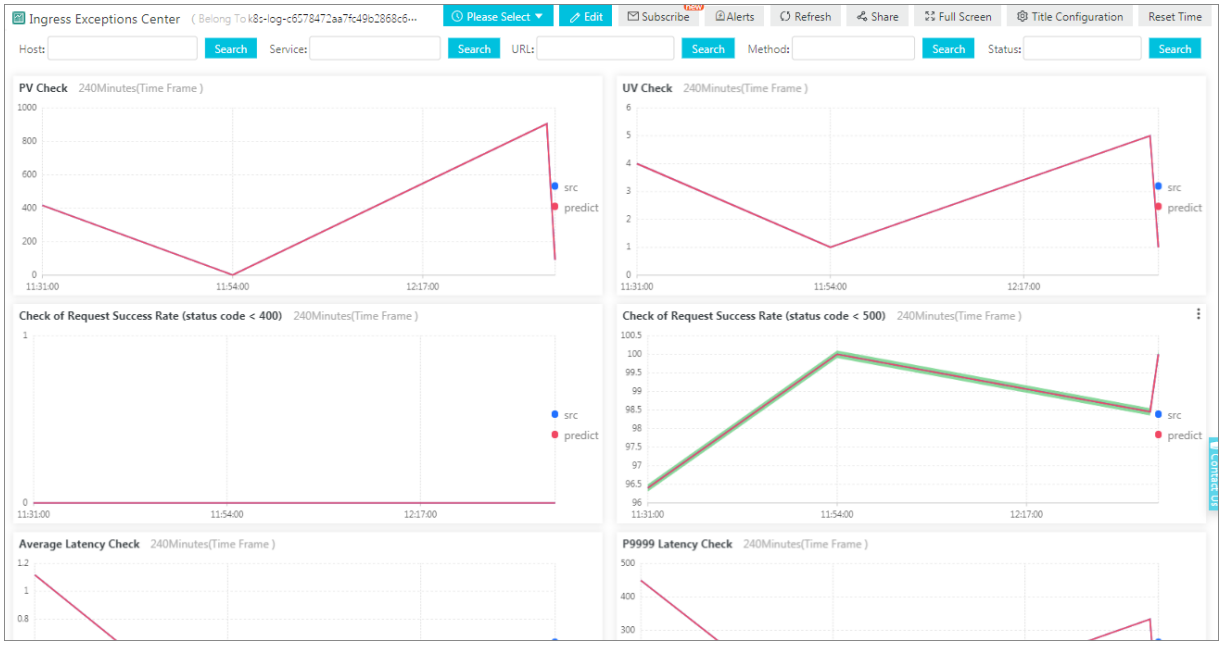
Ingress monitoring center for blue/green deployment

The Ingress monitoring center for blue/green deployment dashboard displays the real-time status of a Service version release and compares the specified Service versions. This allows you to identify exceptions and roll back the Service at the earliest opportunity. You must specify **ServiceA** and **ServiceB** for monitoring and comparison. The dashboard displays the following dynamic monitoring data of each Service: the number of PVs, the proportion of 5xx errors, the success rate of requests, the average latency, the P95, P99, and P999 latencies, and the amount of data transfer.



Ingress exceptions center

The Ingress exceptions center automatically detects anomalies in the log data of nginx-ingress. This service uses the machine learning algorithms provided by Log Service and the time series analysis algorithms.



Configure alerting

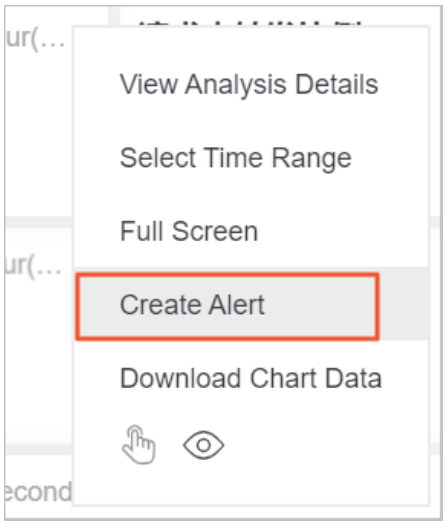
Log Service enables interactive log analytics and provides visualized dashboards. You can also configure alerting based on the dashboards. You can select one or more alerting methods, such as Email, WebHook-DingTalk Bot, WebHook-Custom, or Notifications.

For more information about how to configure alerting, see Create an alert rule.

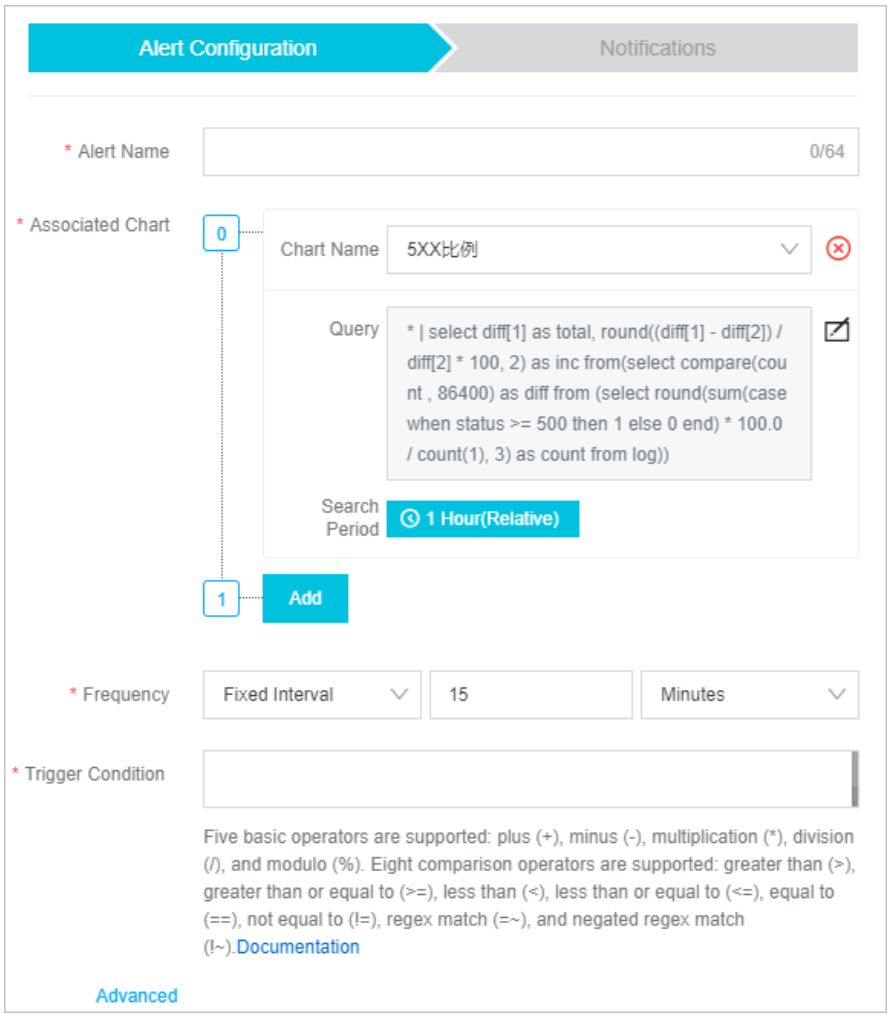
The following example describes how to configure an alert rule for the 5xx Proportion chart. The system checks whether the alert conditions are met every five minutes. If the proportion of 5xx errors within the specified time range reaches 1%, an alert is triggered.

1. In the Dashboard section, click **Ingress Monitoring Center**. On the page that appears, move the

pointer over the ⋮ icon in the upper-right corner of the **5XX Proportion** chart, and click **Create Alert**.



2. On the **Alert Configuration** wizard page, set **Alert Name**, **Search Period**, **Frequency**, and **Trigger Condition**. `total` specified in the Query field is the proportion of 5XX. Set Trigger Condition to `total > 1`.



3. On the **Notifications** wizard page, select one or more alerting methods based on your requirements and set the parameters. Then, click **Submit** to create the alert rule.

Subscribe to a dashboard

Log Service allows you to subscribe to dashboards. This feature takes snapshots of a dashboard and sends the snapshots by email or DingTalk group message at a specified interval.

For more information about how to subscribe to a dashboard, see Subscribe to a dashboard.

The following example describes how to subscribe to the Ingress overview dashboard. After you subscribe to the dashboard, a message is sent at 10:00 every morning to the specified DingTalk group.

1. In the Dashboard section, click **Ingress Overview V1.2**. On the Ingress Overview V1.2 tab, choose **Subscribe > Create**.
2. On the Subscription Configuration wizard page, select *Daily* and *10:00* in the Frequency section. Disable **Add Watermark** and click **Next**.
3. On the Notifications wizard page, set Notifications to **WebHook-DingTalk Bot** and set Request URL to the webhook URL of the DingTalk chatbot. Then, click **Submit** to create the subscription.

6.5.6. Deploy Ingress controllers in high-load scenarios

An Ingress is used to enable Layer 7 load balancing for external access to API objects in Kubernetes clusters. Ingress controllers are used to implement the features of Ingresses. This allows Ingresses to perform load balancing for external access based on Ingress rules. In high-load scenarios, insufficient CPU resources and network connections may downgrade application performance. This topic describes how to improve application performance in high-load scenarios by using Ingress controllers.

Prerequisites

- A Container Service for Kubernetes (ACK) cluster is created. For more information, see 创建Kubernetes托管版集群.
- An Ingress controller runs as normal in the ACK cluster.
- kubectl is installed.

Considerations

Take note of the following items when you deploy the Ingress controller in a high-load scenario.

- Elastic Compute Service (ECS) instance specifications
  When the cluster receives a large number of concurrent requests, Ingresses consume a large amount of CPU resources and network connections. We recommend that you use ECS instance types with enhanced performance, such as:
  - ecs.c6e.8xlarge (32 Core - 64 GB): compute optimized instance type with enhanced performance. This instance type supports up to 6,000,000 packets per second (PPS).
  - ecs.g6e.8xlarge (32 Core - 128 GB): general purpose instance type with enhanced performance. This instance type supports up to 6,000,000 packets per second (PPS).
- Kubernetes configurations
  - Use exclusive nodes to deploy the Ingress controller. Run the following commands to add labels and taints to the nodes:

    ```
    kubectl label nodes $node_name ingress-pod="yes"
    kubectl taint nodes $node_name ingress-pod="yes":NoExecute
    ```

  - Set CPU Policy to `static`.
  - We recommend that you select Super I (slb.s3.large) as the Server Load Balancer (SLB) specification for the ingress-controller Service.

- We recommend that you use Terway as the network plug-in and use the exclusive ENI mode.

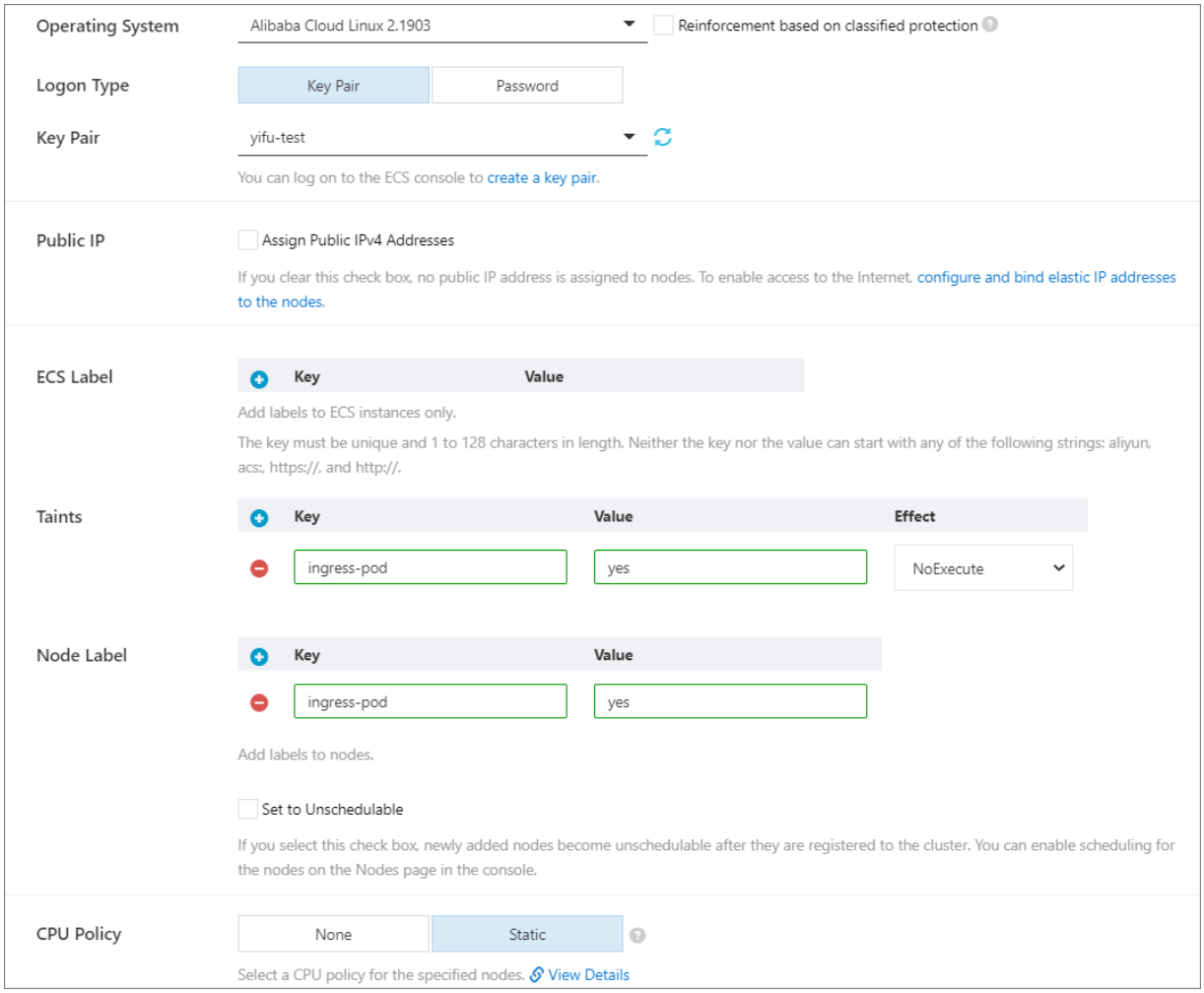
- Ingress controller configurations
  - Configure guaranteed pods for the Ingress controller.
    - Set the `requests` and `limits` parameters of the nginx-ingress-controller containers to 15 Core and 20 GiB.
    - Set the `requests` and `limits` parameters of the init-sysctl init container to 100 m (100 millicore) and 70 MiB.
  - Delete the `podAntiAffinity` parameters from the configurations of the Ingress controller pods. This way, a node can host two Ingress controller pods.
  - Set the number of the pod replicas of the Ingress controller Deployment to a value that is twice the number of newly added nodes.
  - Set `worker-processes` in the ConfigMap of the Ingress controller to 15. This reserves 15 worker processes for the system.
  - Set `keepalive` in the ConfigMap of the Ingress controller to specify the maximum number of requests through a connection.
  - Disable logging.

Step 1: Add nodes

Create a node pool in the ACK cluster and add two nodes to the node pool.

Configure the node pool based on the following description. For more information, see 管理节点池.

- Set **Operating System** to **Alibaba Cloud Linux 2.1903**.
- Set **Node Label** and **Taints**.
  - Add a **taint**. Set **Key** to *ingress-pod*, set **Value** to *yes*, and set **Effect** to **NoExecute**.
  - Add a **node label**. Set **Key** to *ingress-pod* and set **Value** to *yes*.
- Set **CPU Policy** to **Static**.



Step 2: Configure the Ingress controller

Run the `kubectl edit deploy nginx-ingress-controller -n kube-system` command to edit the configuration file of the Ingress controller based on the following description.

- Delete the pod anti-affinity settings.

```
podAntiAffinity:
 requiredDuringSchedulingIgnoredDuringExecution:
 - labelSelector:
    matchExpressions:
    - key: app
    operator: In
    values:
    - ingress-nginx
  topologyKey: kubernetes.io/hostname
```

- Set the `requests` and `limits` parameters for the init container.

```
resources:
 limits:
  cpu: 100m
  memory: 70Mi
 requests:
  cpu: 100m
  memory: 70Mi
```

- Set the `requests` and `limits` parameters of the nginx-ingress-controller containers to 15 Core and 20 GiB.

```
resources:
 limits:
  cpu: "15"
  memory: 20Gi
 requests:
  cpu: "15"
  memory: 20Gi
```

- Set node affinity and tolerations.

```
nodeSelector:
 ingress-pod: "yes"
tolerations:
- effect: NoExecute
 key: ingress-pod
 operator: Equal
 value: "yes"
```

- Set the number of the pod replicas of the Ingress controller Deployment to a value that is twice the number of the newly added nodes.
- Disable metric collection by adding `--enable-metrics=false` to the startup parameters.

> **Note** If you do not need metrics, we recommend that you disable metric collection.

```
containers:
- args:
 - /nginx-ingress-controller
 - --configmap=$(POD_NAMESPACE)/nginx-configuration
 - --tcp-services-configmap=$(POD_NAMESPACE)/tcp-services
 - --udp-services-configmap=$(POD_NAMESPACE)/udp-services
 - --annotations-prefix=nginx.ingress.kubernetes.io
 - --publish-service=$(POD_NAMESPACE)/nginx-ingress-lb
 - --enable-metrics=false
 - --v=1
```

Step 3: Edit the ConfigMap of the Ingress controller

1. Run the `kubectl edit cm nginx-ingress-controller -n kube-system` command to edit the ConfigMap of the Ingress controller. Modify the ConfigMap based on the following template:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-configuration
  namespace: kube-system
data:
  allow-backend-server-header: "true"
  enable-underscores-in-headers: "true"
  generate-request-id: "true"
  ignore-invalid-headers: "true"
  log-format-upstream: $remote_addr - [$remote_addr] - $remote_user [$time_local] "$request" $status $bo
dy_bytes_sent "$http_referer" "$http_user_agent" $request_length $request_time [$proxy_upstream_nam
e] $upstream_addr $upstream_response_length $upstream_response_time $upstream_status $req_id $hos
t [$proxy_alternative_upstream_name]
  max-worker-connections: "65536"
  proxy-body-size: 20m
  proxy-connect-timeout: "3"
  proxy-read-timeout: "5"
  proxy-send-timeout: "5"
  reuse-port: "true"
  server-tokens: "false"
  ssl-redirect: "false"
  upstream-keepalive-timeout: "900"
  worker-processes: 15
  worker-cpu-affinity: auto
  upstream-keepalive-connections: "300"
  upstream-keepalive-requests: "1000"
  keep-alive: "900"
  keep-alive-requests: "10000"
  disable-access-log: "true"
```

> **Note**
> - worker-processes: specifies the number of NGINX processes to be started.
> - keep-alive-requests: specifies the maximum number of requests through a connection.
> - disable-access-log: disables logging.

2. Import logs to a file and set up log rotation.

   By default, logs are written into the */dev/stdout* file. When the cluster receives a large number of requests, the CPU usage is high. In this case, we recommend that you write logs into the /dev/stdout file and set up log rotation.

   i. Use SSH to log on to the ECS instance where the ingress-controller pods are deployed. For more information, see Log on to a Linux server by using SSH.

   ii. Add the following content to the end of the */etc/crontab* file:

   ```
   */15 * * * * root /root/nginx-log-rotate.sh
   ```

   > **Note** In this example, the logs are rotated every 15 minutes. You can change the interval based on your requirements.

iii. Create a file named *nginx-log-rotate.sh* in the */root* directory.

```
#! /bin/bash
# Specify the maximum number of log files that are retained. You can change the number based on your requirements.
keep_log_num=5
ingress_nginx_container_ids=$(docker ps | grep nginx-ingress-controller  | grep -v pause | awk '{print $1}')
if [[ -z "$ingress_nginx_container_ids" ]]; then
  echo "error: failed to get ingress nginx container ids"
  exit 1
fi
# Make the Ingress controller pods sleep for a random length of time period between 5 to 10 seconds.
sleep $(( RANDOM % (10 - 5 + 1 ) + 5 ))
for id in $ingress_nginx_container_ids; do
  docker exec $id  bash -c "cd /var/log/nginx; if [[ \$(ls access.log-* | wc -l) -gt $keep_log_num ]]; then rm -f \$(ls -t access.log-* | tail -1); fi ; mv access.log  access.log-\$(date +%F:%T) ; kill -USR1 \$(cat /tmp/nginx.pid)"
done
```

iv. Run the following command to make the *nginx-log-rotate.sh* file executable:

```
chmod 755 /root/nginx-log-rotate.sh
```

6.5.7. Deploy Ingresses in a high-reliability architecture

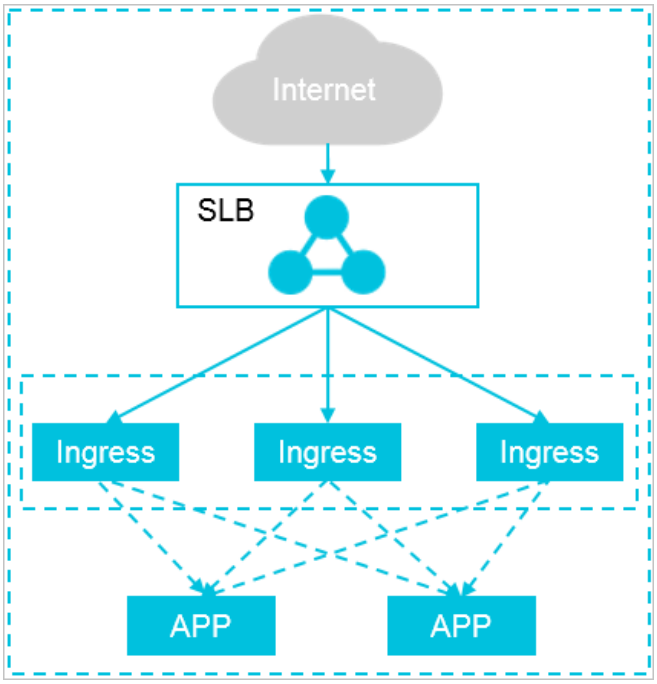
An Ingress is a set of rules that authorize external access to Services within a Kubernetes cluster. Ingresses provide Layer 7 load balancing. You can configure Ingresses to specify the URLs, Server Load Balancer (SLB) instances, Secure Sockets Layer (SSL) connections, and name-based virtual hosts that allow external access. The high reliability of Ingresses is important because Ingresses manage external access to Services within a cluster. This topic describes how to deploy Ingresses in a high-reliability architecture for a Container Service for Kubernetes (ACK) cluster.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- You are connected to the master node of the ACK cluster through SSH. For more information, see Use SSH to connect to an ACK cluster.

High-reliability deployment architecture

A multi-replica deployment architecture is a common solution to provide high reliability and resolve single point of failures (SPOFs). In ACK clusters, Ingresses are deployed in a multi-node architecture to ensure the high reliability of the access layer. Ingresses are the entrance to Kubernetes Services. We recommend that you use exclusive Ingress nodes to avoid scenarios where applications and Ingresses compete for resources.



The access layer in the preceding figure is composed of multiple exclusive Ingress nodes. You can also scale the number of Ingress nodes based on the traffic volume to the backend applications. If your cluster size is small, you can deploy Ingresses and applications together. In this case, we recommend that you isolate resources and restrict resource consumption.

Query the pods of the NGINX Ingress controller and the public IP address of the SLB instance

After an ACK cluster is created, the NGINX Ingress controller with two pods is automatically deployed. An Internet-facing SLB instance is also created as the frontend load balancing Service.

1. To query the pods that are provisioned for the NGINX Ingress controller, run the following command:

   ```
   kubectl -n kube-system get pod | grep nginx-ingress-controller
   ```

   Expected output:

   ```
   nginx-ingress-controller-8648ddc696-2bshk          1/1   Running  0     3h
   nginx-ingress-controller-8648ddc696-jvbs9          1/1   Running  0     3h
   ```

2. To query the public IP address of the frontend SLB instance, run the following command:

   ```
   kubectl -n kube-system get svc nginx-ingress-lb
   ```

   Expected output:

   ```
   NAME         TYPE      CLUSTER-IP   EXTERNAL-IP    PORT(S)           AGE
   nginx-ingress-lb  LoadBalancer  172.xx.x.xx  118.xxx.xxx.xx  80:32457/TCP,443:31370/TCP  21d
   ```

Deploy an Ingress access layer with high reliability

When the cluster size grows, you must expand the access layer to ensure high performance and high availability of the access layer. You can use the following methods to expand the access layer:

- **Method 1: Increase the number of pods**
  You can increase the number of pods that are provisioned for the Deployment of the NGINX Ingress controller to expand the access layer.

i. Run the following command to scale the number of pods to 3:

```
kubectl -n kube-system scale --replicas=3 deployment/nginx-ingress-controller
```

Expected output:

```
deployment.extensions/nginx-ingress-controller scaled
```

ii. After you scale the number of pods, run the following command to query the pods that are provisioned for the NGINX Ingress controller:

```
kubectl -n kube-system get pod | grep nginx-ingress-controller
```

Expected output:

```
nginx-ingress-controller-8648ddc696-2bshk          1/1   Running  0     3h
nginx-ingress-controller-8648ddc696-jvbs9          1/1   Running  0     3h
nginx-ingress-controller-8648ddc696-xqmfn          1/1   Running  0     33s
```

- **Method 2: Deploy Ingresses on nodes with higher specifications**
  You can add labels to nodes with higher specifications. Then, pods provisioned for the NGINX Ingress controller are scheduled to these nodes.

  i. To query information about the nodes in the cluster, run the following command:

  ```
  kubectl get node
  ```

  Expected output:

  ```
  NAME                STATUS  ROLES   AGE  VERSION
  cn-hangzhou.i-bp11bcmsna8d4bp****  Ready   master  21d  v1.11.5
  cn-hangzhou.i-bp12h6biv9bg24l****  Ready   <none>  21d  v1.11.5
  cn-hangzhou.i-bp12h6biv9bg24l****  Ready   <none>  21d  v1.11.5
  cn-hangzhou.i-bp12h6biv9bg24l****  Ready   <none>  21d  v1.11.5
  cn-hangzhou.i-bp181pofzyyksie****  Ready   master  21d  v1.11.5
  cn-hangzhou.i-bp1cbsg6rf3580z****  Ready   master  21d  v1.11.5
  ```

  ii. Run the following command to add the `node-role.kubernetes.io/ingress="true"` label to the `cn-hangzhou.i-bp12h6biv9bg24lmdc2o` and `cn-hangzhou.i-bp12h6biv9bg24lmdc2p` nodes.

  ```
  kubectl label nodes cn-hangzhou.i-bp12h6biv9bg24lmdc2o node-role.kubernetes.io/ingress="true"
  ```

  Expected output:

  ```
  node/cn-hangzhou.i-bp12h6biv9bg24lmdc2o labeled
  ```

  ```
  kubectl label nodes cn-hangzhou.i-bp12h6biv9bg24lmdc2p node-role.kubernetes.io/ingress="true"
  ```

  Expected output:

  ```
  node/cn-hangzhou.i-bp12h6biv9bg24lmdc2p labeled
  ```

> **Note**
>
> - The number of labeled nodes must be no less than the number of pods that are provisioned for the NGINX Ingress controller. This ensures that each pod runs on an exclusive node.
> - If the value of ROLES is *none* in the returned response, it indicates that the related node is a worker node.
> - We recommend that you add labels to and deploy Ingresses on worker nodes.

iii. Run the following command to update the related Deployment by adding the nodeSelector field:

```
kubectl -n kube-system patch deployment nginx-ingress-controller -p '{"spec": {"template": {"spec": {"nodeSelector": {"node-role.kubernetes.io/ingress": "true"}}}}}'
```

Expected output:

```
deployment.extensions/nginx-ingress-controller patched
```

iv. Run the following command to query the pods that are provisioned for the NGINX Ingress controller. The result indicates that pods are scheduled to the nodes that have the `node-role.kubernetes.io/ingress="true"` label.

```
kubectl -n kube-system get pod -o wide | grep nginx-ingress-controller
```

Expected output:

```
nginx-ingress-controller-8648ddc696-2bshk          1/1   Running  0     3h   172.16.2.15   cn-hangzhou.i-bp12h6biv9bg24lmdc2p  <none>
nginx-ingress-controller-8648ddc696-jvbs9          1/1   Running  0     3h   172.16.2.145  cn-hangzhou.i-bp12h6biv9bg24lmdc2o  <none>
```

6.5.8. Configure an Ingress controller to use an internal-facing SLB instance

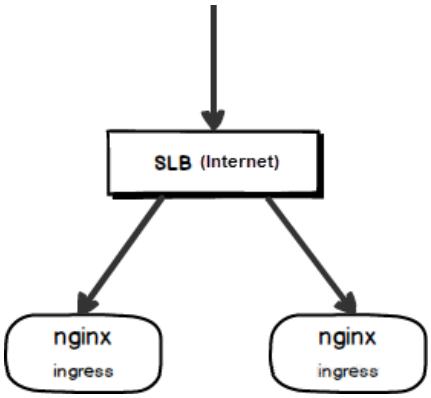
You can configure Container Service for Kubernetes (ACK) clusters to allow access from the Internet and access from other services in the same virtual private cloud (VPC). This topic describes how to configure an Ingress controller to use an internal-facing Server Load Balancer (SLB) instance.

Prerequisites

- A Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群.
- You are connected to a master node of the cluster by using SSH. For more information, see Use SSH to connect to an ACK cluster.

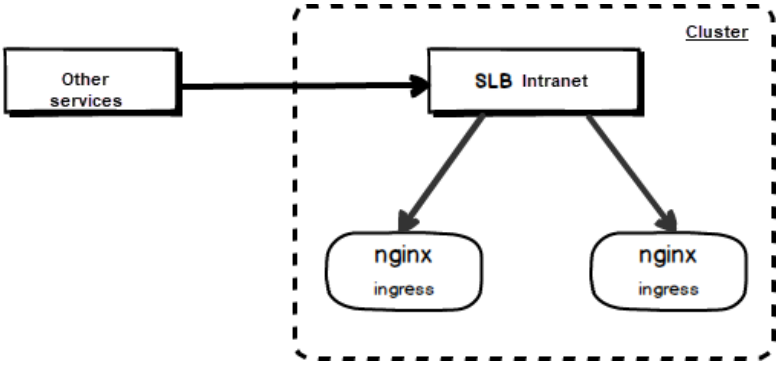
Context

When you create a Kubernetes cluster by using the Container Service console, the system automatically deploys an NGINX Ingress controller in the cluster and associates it with an Internet-facing SLB instance during cluster initialization.



Configure an internal-facing SLB instance

You can modify the configuration of the NGINX Ingress controller to make the cluster accessible to only services that are deployed in the same VPC.



1. Create an internal-facing SLB instance. For more information, see Create a CLB instance.

   > **Note** Create an SLB instance in the same VPC as the cluster. Choose the instance type based on your requirements.

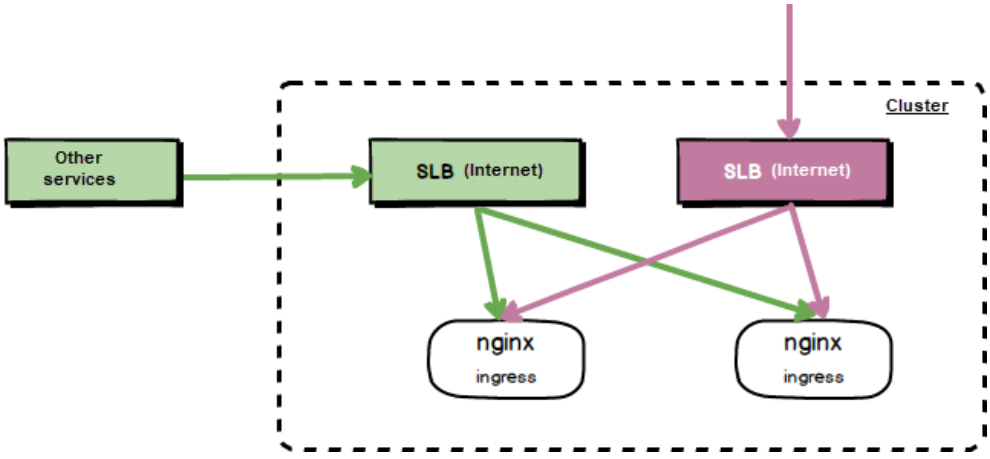
2. Configure the NGINX Ingress controller.
   After you have created an internal-facing SLB instance, configure the NGINX Ingress controller to use the SLB instance with the following annotations. For more information, see Use annotations to configure load balancing.

```
# nginx ingress slb service
apiVersion: v1
kind: Service
metadata:
 name: nginx-ingress-lb
 namespace: kube-system
 labels:
  app: nginx-ingress-lb
 annotations:
  # Specify that the SLB instance uses an internal IP address
  service.beta.kubernetes.io/alicloud-loadbalancer-address-type: intranet
  # Specify the ID of the created internal-facing SLB instance.
  service.beta.kubernetes.io/alicloud-loadbalancer-id: <YOUR_INTRANET_SLB_ID>
  # Specify whether to automatically create listeners, which overwrite existing listeners. You can also manual
ly create listeners.
  #service.beta.kubernetes.io/alicloud-loadbalancer-force-override-listeners: 'true'
spec:
 type: LoadBalancer
 # route traffic to other nodes
 externalTrafficPolicy: "Cluster"
 ports:
 - port: 80
  name: http
  targetPort: 80
 - port: 443
  name: https
  targetPort: 443
 selector:
  # select app=ingress-nginx pods
  app: ingress-nginx
```

After the configuration is applied, the NGINX Ingress controller (kube-system/nginx-ingress-lb) uses the newly specified internal-facing SLB instance.

Use an internal-facing SLB instance and an Internet-facing SLB instance together

In some scenarios, you may want the cluster to allow access from the Internet and access from other services in the same VPC at the same time. To do this, you need only to deploy another NGINX Ingress controller (for example, kube-system/nginx-ingress-lb-intranet) in the cluster.



> **Note** By default, a kube-system/nginx-ingress-lb Ingress controller is created during cluster initialization. This Ingress controller uses an Internet-facing SLB instance.

1. Create an internal-facing SLB instance. For more information, see Create a CLB instance.

   > **Note** Create an SLB instance in the same VPC as the cluster. Choose the instance type based on your requirements.

2. Create a new NGINX Ingress controller.
   After you have created an internal-facing SLB instance, you can use the following YAML file to create a kube-system/nginx-ingress-lb-intranet service.

```
# intranet nginx ingress slb service
apiVersion: v1
kind: Service
metadata:
 # Set the service name to nginx-ingress-lb-intranet.
 name: nginx-ingress-lb-intranet
 namespace: kube-system
 labels:
  app: nginx-ingress-lb-intranet
 annotations:
  # Specify that the SLB instance uses an internal IP address
  service.beta.kubernetes.io/alicloud-loadbalancer-address-type: intranet
  # Specify the ID of the created internal-facing SLB instance.
  service.beta.kubernetes.io/alicloud-loadbalancer-id: <YOUR_INTRANET_SLB_ID>
  # Specify whether to automatically create listeners, which overwrite existing listeners. You can also manual
ly create listeners.
  #service.beta.kubernetes.io/alicloud-loadbalancer-force-override-listeners: 'true'
spec:
 type: LoadBalancer
 # route traffic to other nodes
 externalTrafficPolicy: "Cluster"
 ports:
 - port: 80
  name: http
  targetPort: 80
 - port: 443
  name: https
  targetPort: 443
 selector:
  # select app=ingress-nginx pods
  app: ingress-nginx
```

After the kube-system/nginx-ingress-lb-intranet service is created, run the `kubectl -n kube-system get svc | grep nginx-ingress-lb` command and verify that two NGINX Ingress controllers are running. One is associated with an Internet-facing SLB instance, and the other with an internal-facing SLB instance.

```
kubectl -n kube-system get svc | grep nginx-ingress-lb
```

```
nginx-ingress-lb        LoadBalancer  172.1*.*.**   47.96.2**.**  80:31456/TCP,443:30016/TCP  5h
nginx-ingress-lb-intranet  LoadBalancer  172.19.*.***  192.16*.*.**  80:32394/TCP,443:31000/TCP  7m
```

When you expose services through Ingresses, you can allow Internet access through the Internet-facing SLB instance and also access from other services in the same VPC network through the internal-facing SLB instance.

6.5.9. Deploy multiple ingress controllers on a cluster

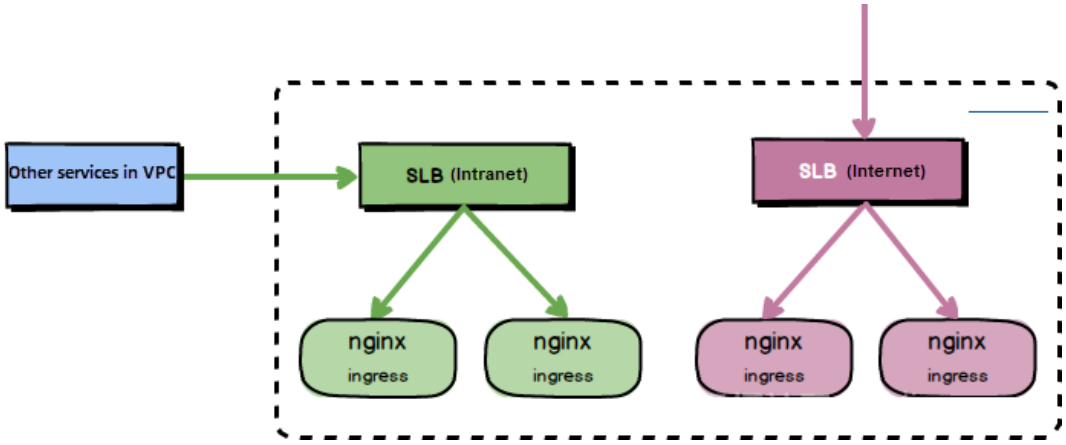
This topic describes how to deploy multiple independent NGINX ingress controllers on a cluster to provide different services for external users.

Prerequisites

- 创建Kubernetes托管版集群
- Use SSH to connect to an ACK cluster

Context

The Configure an Ingress controller to use an internal-facing SLB instance topic describes how to modify the default configurations of the NGINX ingress controller to use internal SLB instances. The two solutions described in the topic meet the needs of most scenarios. Assume that you have deployed multiple services on a cluster. This topic describes a solution that applies to scenarios where some services must be available to external users, but other services only allow requests from non-Kubernetes workloads in the same virtual private cloud (VPC). To meet such demands, you can deploy two independent NGINX ingress controllers and bind them to SLB instances of different network types.



Deploy a second NGINX ingress controller

You can perform the following steps to deploy an independent NGINX ingress controller to a cluster.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **Alibaba Cloud Apps** tab, click **ack-ingress-nginx**.
4. On the **App Catalog - ack-ingress-nginx** page, click the **Parameters** tab. In the **Deploy** section, specify a cluster, a namespace, and a release name, and click **Create**.

   The following table describes the parameters.

Parameter	Description
controller.image.repository	The image repository of ingress-nginx. If the cluster is deployed in a region outside China, we recommend that you set this parameter to the region ID.

Parameter	Description
controller.image.tag	The image version of ingress-nginx. For more information, see NGINX Ingress controller.
controller.ingressClass	The ingress class of the ingress controller. The ingress controller handles only the ingress resources that are annotated with the ingress class. **Notice** In a cluster, the ingress class of each ingress controller must be unique. The ingress class of the default ingress controller in a cluster is nginx. Therefore, do not set this parameter to nginx.
controller.replicaCount	The number of pod replicas of the ingress controller.
controller.service.enabled	Specifies whether to use a public-facing SLB instance for load balancing. If you do not want to use a public-facing SLB instance, set the value to false.
controller.service.internal.enabled	Specifies whether to use an internal SLB instance for load balancing. If you want to use an internal SLB instance, set this parameter to true.
controller.kind	The deployment mode of the ingress controller. Valid values: Deployment and DaemonSet.

In the left-side navigation pane of the ACK console, click **Clusters**. On the Clusters page, find the cluster that you want to view, and click **Applications** in the **Actions** column. The details page of the cluster appears. In the left-side navigation pane, click **Releases**. On the **Helm** tab, view the newly deployed ingress controller.

Test the connectivity

The following example deploys a test application and uses the newly deployed NGINX ingress controller to provide application services for external users.

1. Deploy an NGINX application.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   run: nginx
 template:
  metadata:
   labels:
    run: nginx
  spec:
   containers:
   - image: nginx
    imagePullPolicy: Always
    name: nginx
    ports:
    - containerPort: 80
     protocol: TCP
   restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
 name: nginx
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 sessionAffinity: None
 type: NodePort
```

2. Use the NGINX ingress controller to provide application services for external users.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: nginx
 annotations:
  # Set the value to the ingress class of the new NGINX ingress controller.
  kubernetes.io/ingress.class: "<YOUR_INGRESS_CLASS>"
spec:
 rules:
 - host: foo.bar.com
  http:
   paths:
   - path: /
    backend:
     serviceName: nginx
     servicePort: 80
```

> **Note** You must configure the kubernetes.io/ingress.class annotation.

After you deploy the application, the ingress IP address is the same as the IP address of the new NGINX ingress controller.

```
kubectl -n kube-system get svc nginx-ingress-lb
```

```
NAME         TYPE       CLUSTER-IP   EXTERNAL-IP   PORT(S)             AGE
nginx-ingress-lb  LoadBalancer  172.19.7.30  47.95.97.115  80:31429/TCP,443:32553/TCP  2d
```

```
kubectl -n <YOUR_NAMESPACE> get svc nginx-ingress-lb
```

```
NAME         TYPE       CLUSTER-IP    EXTERNAL-IP    PORT(S)             AGE
nginx-ingress-lb  LoadBalancer  172.19.6.227  39.105.252.62  80:30969/TCP,443:31325/TCP  39m
```

```
kubectl get ing
```

```
NAME    HOSTS     ADDRESS      PORTS   AGE
nginx   foo.bar.com  39.105.252.62  80     5m
```

3. Access the application through the default and new NGINX ingress controllers.

```
 # Access the application through the default NGINX ingress controller. The 404 status code is expected.
 curl -H "Host: foo.bar.com" http://47.95.97.115
default backend - 404
 # Access the application through the new NGINX ingress controller. The NGINX welcome page is expected. cu
rl -H "Host: foo.bar.com" http://39.105.252.62
<! DOCTYPE html>
<html>
<head>
<title>Welcome to nginx! </title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx! </h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

The preceding tests show that application services accessed through different NGINX ingress controllers do not interfere each other. This solution applies to scenarios where some services must be available to external users, but other services only allow requests from non-Kubernetes workloads in the same VPC.

Related information

- Deploy Ingresses in a high-reliability architecture
- Configure an Ingress controller to use an internal-facing SLB instance

6.5.10. Use an Ingress controller to mirror network traffic

Container Service for Kubernetes (ACK) allows you to mirror network traffic to different clusters. Traffic mirroring is used in system simulations and troubleshooting. This topic describes how to use an Ingress controller to mirror network traffic.

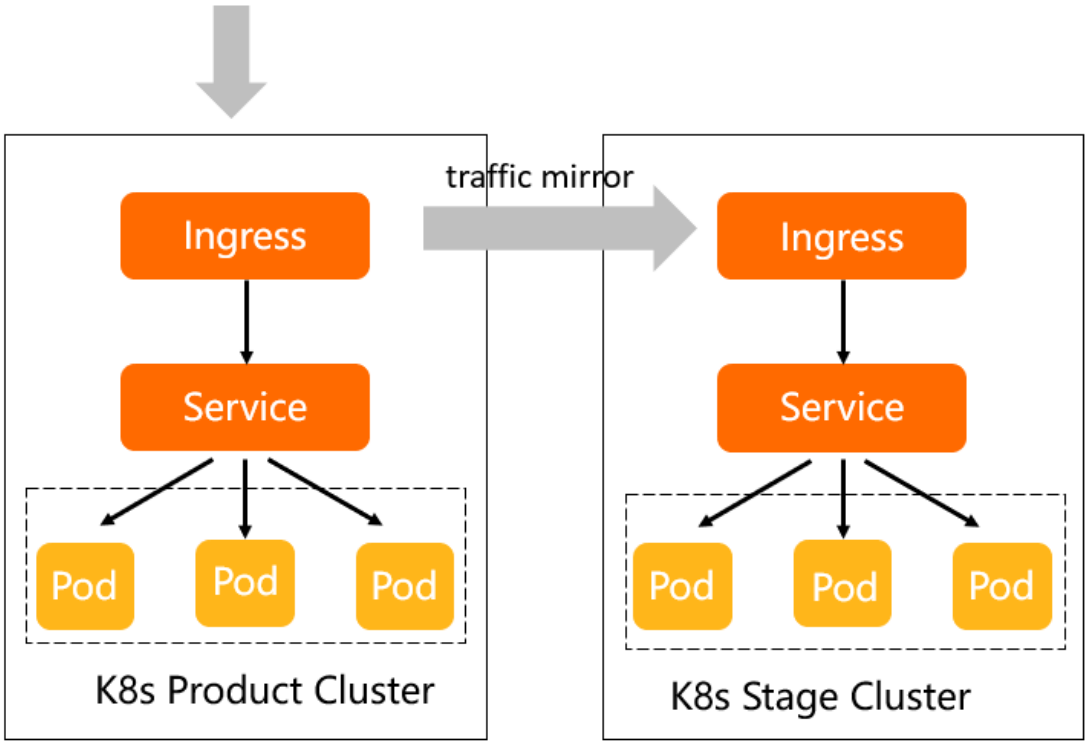
Prerequisites

Two ACK clusters are created. For more information, see 创建Kubernetes托管版集群. In this topic, a cluster named K8s Product Cluster is used as the production environment and another cluster named K8s Stage Cluster is used as the staging environment.

Scenarios

Traffic mirroring can be used in the following scenarios:

- Major changes are to be made to a system or new features are to be released. In this case, you must run stress tests to evaluate the stability of the system. Production workloads are usually simulated in a staging environment to test the stability of a new system before the system is released. However, the actual loads are difficult to estimate because the system may receive both normal and abnormal network traffic. To address this issue, you can mirror network traffic from applications that are deployed in the production environment to the staging environment. Then, you can simulate the production workloads in the staging environment.
- When a system deployed in the production environment encounters a performance bottleneck and you cannot locate the cause, you can mirror the network traffic of the system to a staging environment. This way, you can troubleshoot errors in the staging environment.



Step 1: Deploy basic applications

1. Deploy applications in the cluster named K8s Product Cluster and check whether the applications are accessible.
   i. Deploy an application in the cluster named K8s Product Cluster and use an Ingress to expose the Services in the cluster.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   # The previous version of the image is deployed.
   - image: registry.cn-hangzhou.aliyuncs.com/xianlu/old-nginx
     imagePullPolicy: Always
     name: nginx
     ports:
     - containerPort: 80
       protocol: TCP
   restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
 name: nginx-service
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: NodePort
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: nginx-ingress
spec:
 rules:
 # By default, the application is deployed in the region of the cluster. You can specify a region and config
ure DNS resolution.
 - host: nginx.c37bf6b77bded43669ba2fb67448b****.cn-hangzhou.alicontainer.com
  http:
   paths:
   - path: /
    backend:
     serviceName: nginx-service
     servicePort: 80
```

ii. Run the following command to view the Ingress settings:

```
kubectl get ing nginx-ingress
```

Expected output:

```
NAME        HOSTS                                        ADDRESS     PORTS   AGE
nginx-ingress   nginx.c37bf6b77bded43669ba2fb67448b****.cn-hangzhou.alicontainer.com   47.110.1**.**   80      8m
```

iii. Run the following command to check whether you can access the domain name of the application:

```
curl http://nginx.c37bf6b77bded43669ba2fb67448b****.cn-hangzhou.alicontainer.com
```

2. Deploy the same application in the cluster named K8s Stage Cluster and check whether the application is accessible.

i. Deploy the same application in the cluster named K8s Stage Cluster and use an Ingress to expose the Services in the cluster.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   # The latest version of the image is deployed.
   - image: registry.cn-hangzhou.aliyuncs.com/xianlu/new-nginx
     imagePullPolicy: Always
     name: nginx
     ports:
     - containerPort: 80
       protocol: TCP
   restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
 name: nginx-service
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: nginx
 type: NodePort
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: nginx-ingress
spec:
 rules:
 # By default, the application is deployed in the region of the cluster. You can specify a region and config
ure DNS resolution.
 - host: nginx.c41eb6ca34a3e49f7aea63b8bc9e8****.cn-beijing.alicontainer.com
  http:
   paths:
   - path: /
    backend:
     serviceName: nginx-service
     servicePort: 80
```

ii. Run the following command to view the Ingress settings:

```
kubectl get ing nginx-ingress
```

Expected output:

```
NAME      HOSTS                                                ADDRESS     PORTS   AGE
nginx-ingress   nginx.c41eb6ca34a3e49f7aea63b8bc9e8****.cn-beijing.alicontainer.com   39.106.***.*   80
1m
```

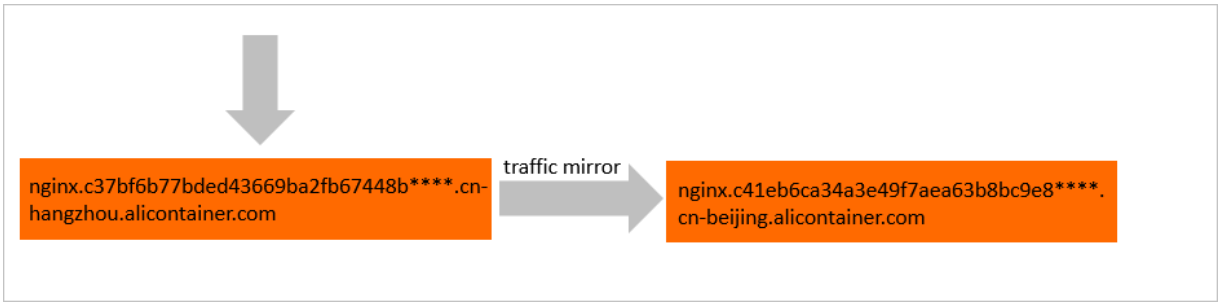
iii. Run the following command to check whether you can access the domain name of the application:

```
curl http://nginx.c41eb6ca34a3e49f7aea63b8bc9e8****.cn-beijing.alicontainer.com
```

Step 2: Mirror the network traffic

Mirror all network traffic of the application deployed in the cluster K8s Product Cluster to Services in the cluster K8s Stage Cluster. This way, all requests that are sent to the domain name `nginx.c37bf6b77bded43669ba2fb67448b****.cn-hangzhou.alicontainer.com` are replicated to the domain name `nginx.c41eb6ca34a3e49f7aea63b8bc9e8****.cn-beijing.alicontainer.com`.

Note The Ingress of the cluster K8s Stage Cluster functions only as a receptor to receive the mirrored network traffic. Do not modify the settings of the Ingress in the cluster K8s Stage Cluster. You can modify the Ingress of the cluster K8s Product Cluster.



1. Run the following command to modify *nginx-configuration*. Set the percentage of network traffic that you want to mirror and specify the domain name that receives the mirrored network traffic:

   ```
   kubectl -n kube-system edit cm nginx-configuration
   ```

   Add the following content to *nginx-configuration*:

   ```
   http-snippet: |
     split_clients "$date_gmt" $mirror_servers {
       100%   nginx.c41eb6ca34a3e49f7aea63b8bc9e8****.cn-beijing.alicontainer.com;
     }
   # Notes on configurations:
   # 1. The percentage value of network traffic that is mirrored must be greater than 0 and can be at most 100. T
   he sum of percentage values cannot exceed 100.
   # 2. You can set one or more applications that receive the mirrored network traffic.
   ```

2. Run the following command to modify *nginx-ingress*:

   You can add traffic mirroring settings to the source Ingress by setting the configuration-snippet and server-snippet annotations.

   ```
   kubectl edit ingress nginx-ingress
   ```

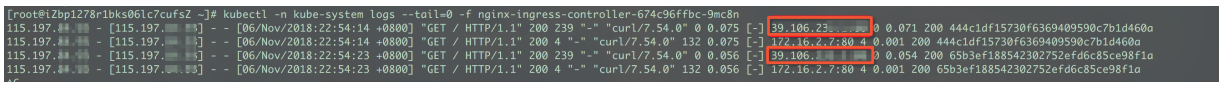
Modify `configuration-snippet` and `server-snippet` of the nginx-ingress controller based on your business requirements. The following YAML file is an example:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: nginx-ingress
 annotations:
  nginx.ingress.kubernetes.io/configuration-snippet: |
    mirror /mirror; # You can repeat this setting N times to mirror the network traffic N times.
  nginx.ingress.kubernetes.io/server-snippet: |
    location = /mirror {
      internal;
      # Does not print the log of mirrored requests
      #access_log off;
      # Set proxy_upstream_name in the following format: [Namespace]-[BackendServiceName]-[BackendSe
rvicePort]
      set $proxy_upstream_name   "default-nginx-service-80";
      # A custom string that is carried in the request header X-Shadow-Service and then passed to the mirror
server.
      set $shadow_service_name   "nginx-product-service";
      proxy_set_header X-Shadow-Service  $shadow_service_name;
      proxy_set_header Host $mirror_servers;
      proxy_pass http://$mirror_servers$request_uri;
    }
spec:
 rules:
 - host: nginx.c37bf6b77bded43669ba2fb67448b****.cn-hangzhou.alicontainer.com
  http:
   paths:
   - path: /
    backend:
     serviceName: nginx-service
     servicePort: 80
```

Check the result

Access the domain name `nginx.c37bf6b77bded43669ba2fb67448b****.cn-hangzhou.alicontainer.com`. This domain name belongs to the application that is deployed in the cluster K8S Product Cluster. Run the following command. The result shows that each request sent to the domain name of the application deployed in the cluster K8S Product Cluster is replicated, and then sent to the application deployed in the cluster K8S Stage Cluster.

```
kubectl -n kube-system logs --tail-o -f nginx-ingress-controller-674c96ffbc-9mc8n
```



6.5.11. FAQ about Ingresses

This topic provides answers to some frequently asked questions about Ingresses.

- Which Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol versions are supported by Ingresses?
- Do Ingresses pass Layer 7 request headers?
- Can ingress-nginx forward requests to HTTPS Services at the backend?

- Do Ingresses pass client IP addresses at Layer 7?
- Does nginx-ingress-controller support HTTP Strict Transport Security (HSTS)?
- Which rewrite rules are supported by ingress-nginx?
- How do I fix the issue that Log Service cannot parse logs as expected after ingress-nginx-controller is upgraded?
- How do I configure the internal-facing SLB instance for the NGINX Ingress controller?
- What are the system updates after I upgrade the NGINX Ingress controller on the Add-ons page in the console?

Which Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol versions are supported by Ingresses?

By default, ingress-nginx supports only TLS 1.2. If the TLS protocol version that is used by a browser or mobile client is earlier than TLS 1.2, errors may occur during handshakes between the client and ingress-nginx.

If you want ingress-nginx to support more TLS protocol versions, run the following commands to add required configurations to the `nginx-configuration` ConfigMap in the kube-system namespace. For more information, see TLS/HTTPS.

```
ssl-ciphers: "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA"
ssl-protocols: "TLSv1 TLSv1.1 TLSv1.2 TLSv1.3"
```

Do Ingresses pass Layer 7 request headers?

By default, ingress-nginx passes Layer 7 request headers. However, request headers that do not conform to HTTP rules are filtered out before requests are forwarded to the backend Services. For example, the Mobile Version request header is filtered out. If you do not want to filter out these request headers, run the following command to add the required configurations to the nginx-configuration ConfigMap. For more information, see ConfigMap.

```
enable-underscores-in-headers: true
```

Can ingress-nginx forward requests to HTTPS Services at the backend?

To enable ingress-nginx to forward requests to HTTPS Services at the backend, run the following command to add the required annotations to the Ingress configuration:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: xxxx
 annotations:
  # Note: You must specify the backend protocol as HTTPS to enable ingress-nginx to forward requests to HTTPS Services at the backend.
  nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
```

Do Ingresses pass client IP addresses at Layer 7?

By default, ingress-nginx adds the X-Forward-For and X-Real-IP header fields to carry client IP addresses. However, if the X-Forward-For and X-Real-IP header fields are already added to the request by a client, the backend server cannot obtain the client IP address.

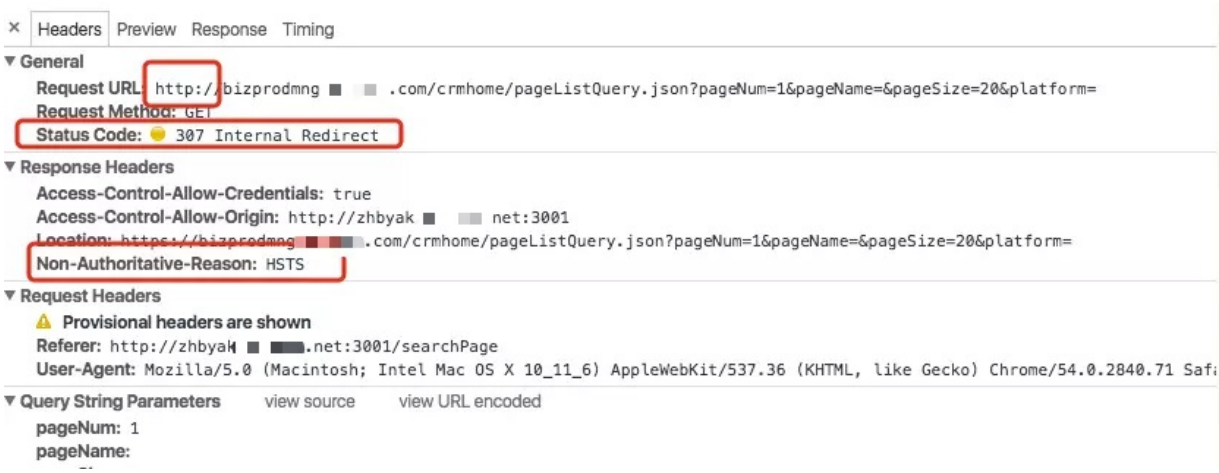
Run the following command to modify the `nginx-configuration` ConfigMap in the kube-system namespace. This allows ingress-nginx to pass client IP addresses at Layer 7.

```
compute-full-forwarded-for: "true"
forwarded-for-header: "X-Forwarded-For"
use-forwarded-headers: "true"
```

For more information, see Configure an ACK Ingress to pass through client IP addresses.

Does nginx-ingress-controller support HTTP Strict Transport Security (HSTS)?

By default, nginx-ingress-controller is enabled for nginx-ingress-controller. When a browser sends a plain HTTP request for the first time, the response header from the backend server (with HSTS enabled) contains `Non-Authoritative-Reason: HSTS`. This indicates that the backend server supports HSTS. If the client also supports HSTS, the client directly sends HTTPS requests after the first successful access. The response body from the backend server contains the status code `307 Internal Redirect`, as shown in the following figure.



If you do not want the client requests to be forwarded to HTTPS Services at the backend, you can disable HSTS for nginx-ingress-controller. For more information, see HSTS.

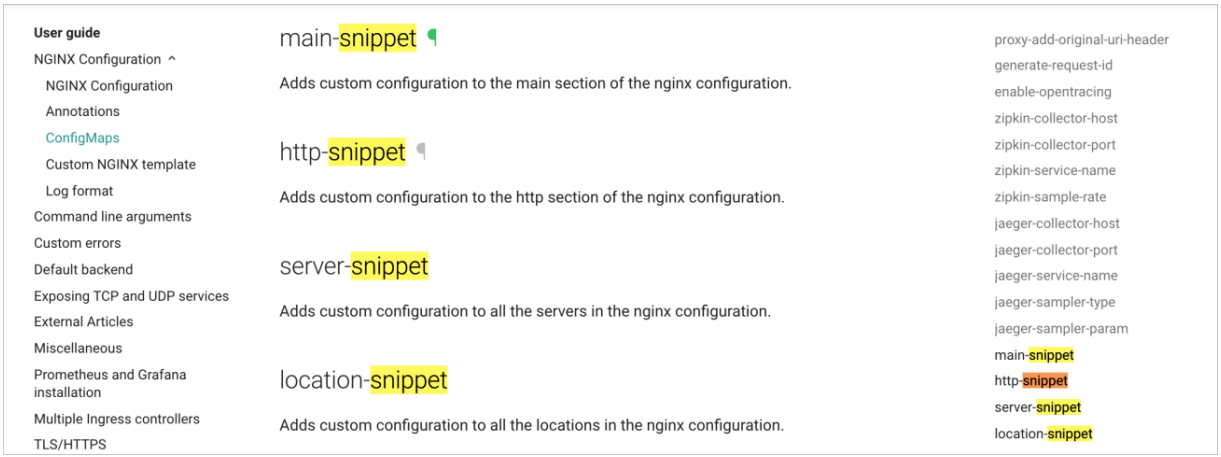
> **Note** By default, the HSTS configuration is cached by browsers. You must manually delete the browser cache after you disable HSTS for nginx-ingress-controller.

Which rewrite rules are supported by ingress-nginx?

Only simple rewrite rules are supported by ingress-nginx. For more information, see Rewrite. If you want to configure complex rewrite rules, use the following methods:

- configuration-snippet: Add this annotation to the location configuration of an Ingress. For more information, see Configuration snippet.
- server-snippet: Add this annotation to the server configuration of an Ingress. For more information, see Server snippet.

You can use other snippets to add global configurations, as shown in the following figure. For more information, see main-snippet.



How do I fix the issue that Log Service cannot parse logs as expected after ingress-nginx-controller is upgraded?

The ingress-nginx-controller component has two commonly used versions: ingress-nginx-controller 0.20 and 0.30. The default log format of ingress-nginx-controller 0.20 is different from that of ingress-nginx-controller 0.30. Therefore, after you upgrade ingress-nginx-controller from 0.20 to 0.30 on the **Add-ons** page in the console, the Ingress dashboard may not show the actual statistics of access to the backend Services when you perform canary releases or blue-green releases with an Ingress.

To fix the issue, perform the following steps to update the `nginx-configuration` ConfigMap and the configuration of `k8s-nginx-ingress`.

1. Update the `nginx-configuration` ConfigMap.
   - If you have not modified the `nginx-configuration` ConfigMap, copy the following content to a file named `nginx-configuration.yaml` and run the `kubectl apply -f nginx-configuration.yaml` command to deploy the file.

```
apiVersion: v1
kind: ConfigMap
data:
 allow-backend-server-header: "true"
 enable-underscores-in-headers: "true"
 generate-request-id: "true"
 ignore-invalid-headers: "true"
 log-format-upstream: $remote_addr - [$remote_addr] - $remote_user [$time_local] "$request" $status $
body_bytes_sent "$http_referer" "$http_user_agent" $request_length $request_time [$proxy_upstream
_name] $upstream_addr $upstream_response_length $upstream_response_time $upstream_status $req
_id $host [$proxy_alternative_upstream_name]
 max-worker-connections: "65536"
 proxy-body-size: 20m
 proxy-connect-timeout: "10"
 reuse-port: "true"
 server-tokens: "false"
 ssl-redirect: "false"
 upstream-keepalive-timeout: "900"
 worker-cpu-affinity: auto
metadata:
 labels:
  app: ingress-nginx
 name: nginx-configuration
 namespace: kube-system
```

- If you have modified the `nginx-configuration` ConfigMap, run the following command to update the configuration. This ensures that your previous modifications are not overwritten.

  ```
  kubectl edit configmap nginx-configuration -n kube-system
  ```

  Add `[$proxy_alternative_upstream_name]` to the end of the `log-format-upstream` field, save the changes, and then exit.

2. Update the configuration of `k8s-nginx-ingress`.

   Copy the following content to a file named `k8s-nginx-ingress.yaml` and run the `kubectl apply -f k8s-nginx-ingress.yaml` command to deploy the file.

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
  namespace: kube-system
 # your config name, must be unique in you k8s cluster
 name: k8s-nginx-ingress
spec:
 # logstore name to upload log
 logstore: nginx-ingress
 # product code, only for k8s nginx ingress
 productCode: k8s-nginx-ingress
 # logtail config detail
 logtailConfig:
  inputType: plugin
  # logtail config name, should be same with [metadata.name]
  configName: k8s-nginx-ingress
  inputDetail:
   plugin:
    inputs:
    - type: service_docker_stdout
     detail:
      IncludeLabel:
       io.kubernetes.container.name: nginx-ingress-controller
      Stderr: false
      Stdout: true
    processors:
    - type: processor_regex
     detail:
      KeepSource: false
      Keys:
      - client_ip
      - x_forward_for
      - remote_user
      - time
      - method
      - url
      - version
      - status
      - body_bytes_sent
      - http_referer
      - http_user_agent
      - request_length
      - request_time
      - proxy_upstream_name
      - upstream_addr
      - upstream_response_length
      - upstream_response_time
      - upstream_status
      - req_id
      - host
      - proxy_alternative_upstream_name
      NoKeyError: true
      NoMatchError: true
      Regex: ^(\S+)\s-\s\[([^]]+)]\s-\s(\S+)\s\[(\S+)\s\S+\s"(\w+)\s(\S+)\s([^"]+)"\s(\d+)\s(\d+)\s"([^"]*)"\s"([^"]*)
"\s(\S+)\s(\S+)+\s\[([^]]*)]\s(\S+)\s(\S+)\s(\S+)\s(\S+)\s(\S+)\s*(\S*)\s*\[*([^]]*)\]*.*
      SourceKey: content
```

What are the system updates after I upgrade the NGINX Ingress controller on the Add-ons page in the console?

If the version of the NGINX Ingress controller is earlier than 0.44, the component includes the following resources:

- serviceaccount/ingress-nginx
- configmap/nginx-configuration
- configmap/tcp-service
- configmap/udp-services
- clusterrole.rbac.authorization.k8s.io/ingress-nginx
- clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx
- role.rbac.authorization.k8s.io/ingress-nginx
- rolebinding.rbac.authorization.k8s.io/ingress-nginx
- service/nginx-ingress-lb
- deployment.apps/nginx-ingress-controller

If the version of the NGINX Ingress controller is 0.44 or later, the component includes the following resources in addition to the previously mentioned resources:

- validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission
- service/ingress-nginx-controller-admission
- serviceaccount/ingress-nginx-admission
- clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission
- clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission
- role.rbac.authorization.k8s.io/ingress-nginx-admission
- rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission
- job.batch/ingress-nginx-admission-create
- job.batch/ingress-nginx-admission-patch

When you upgrade the NGINX Ingress controller on the Add-ons page in the console, the configurations of the following resources are unchanged:

- configmap/nginx-configuration
- configmap/tcp-services
- configmap/udp-services
- service/nginx-ingress-lb

The configurations of other resources are overwritten to default values. For example, the default value of the replicas parameter of deployment.apps/nginx-ingress-controller is 2. If you set the value of replicas to 5 before you upgrade the NGINX Ingress controller, the replicas parameter uses the default value 2 after you upgrade the component on the Add-ons page.

6.6. Service discovery DNS

6.6.1. Overview

CoreDNS is deployed in Container Service for Kubernetes (ACK) clusters and serves as a Domain Name System (DNS) server to provide DNS resolution services for workloads deployed in ACK clusters. This topic introduces CoreDNS and its features, and describes how CoreDNS resolves domain names in Kubernetes clusters.

Introduction to CoreDNS

CoreDNS is a DNS resolver for Kubernetes clusters. CoreDNS resolves custom internal domain names and external domain names. CoreDNS contains a variety of plug-ins. These plug-ins allow you to configure custom DNS settings and customize hosts, Canonical Name (CNAME) records, and rewrite rules for Kubernetes clusters. CoreDNS is hosted by Cloud Native Computing Foundation (CNCF), which also hosts Kubernetes. For more information about CoreDNS, see CoreDNS: DNS and Service Discovery.

ACK uses CoreDNS to implement service discovery in clusters. You can configure CoreDNS based on your requirements to support higher numbers of DNS queries in different scenarios.

- For more information about CoreDNS configurations in ACK clusters, see Configure CoreDNS for an ACK cluster.
- For more information about how to use CoreDNS to improve the performance of DNS resolutions for an ACK cluster, see Optimize DNS resolution for an ACK cluster.

> **Note**
> You can also use NodeLocal DNSCache to improve the stability and performance of service discovery in ACK clusters. NodeLocal DNSCache improves DNS performance by running a DNS caching agent on nodes as a DaemonSet. For more information about how to deploy NodeLocal DNSCache in an ACK cluster, see Use NodeLocal DNSCache in an ACK cluster.

Mappings between CoreDNS versions and Kubernetes versions

ACK maintains a mapping between CoreDNS versions and Kubernetes versions. You can upgrade CoreDNS versions only when you upgrade Kubernetes versions for ACK clusters. The following table describes the mapping between CoreDNS versions and Kubernetes versions.

> **Note** To view the current CoreDNS version of your cluster, log on to the ACK console, go to the **Pods** page, and then check the image version of the pod named coredns-xxx. For more information, see Manage pods.

Item	Version			
Kubernetes	1.12	1.14	1.16	1.18
CoreDNS	1.2.6	1.3.1	1.6.2	1.6.7

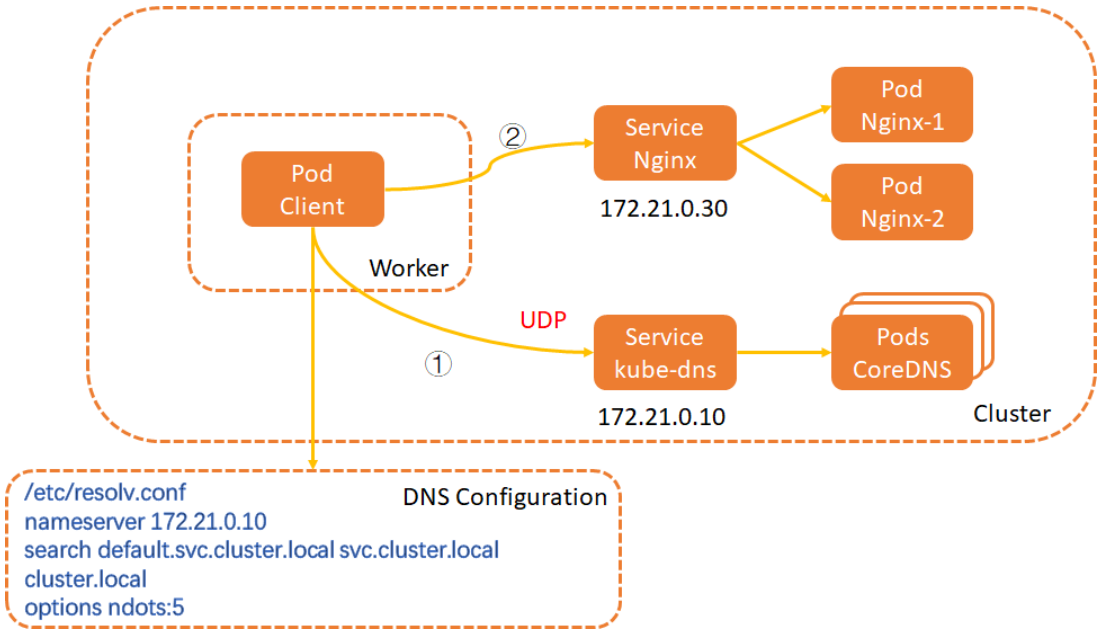
DNS resolutions in an ACK cluster

The startup parameters of kubelet in an ACK cluster include `--cluster-dns=<dns-service-ip>` and `--cluster-domain=<default-local-domain>`. These parameters are used to configure the IP address and the suffix of the base domain name for the DNS server in the ACK cluster.

By default, ACK deploys a set of workloads in a cluster to run CoreDNS. A Service named **kube-dns** is deployed to expose these workloads to DNS queries in the cluster. Two pods named **coredns** are deployed as the backend of CoreDNS. DNS queries in the cluster are sent to the DNS server that is specified in the coredns pod configurations. The DNS configuration file in the pod is */etc/resolv.conf*. The file contains the following content:

```
nameserver 172.xx.x.xx
search kube-system.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

The following figure shows how domain names are resolved in an ACK cluster.



No.	Description
①	When a client pod attempts to access Service Nginx, the pod sends a request to the DNS server that is specified in the DNS configuration file /etc/resolv.conf. In this example, the DNS server address is 172.21.0.10, which is the IP address of Service kube-dns. The result of the resolution is 172.21.0.30.
②	The client pod sends another request to 172.21.0.30, which is the IP address of Service Nginx. Then, the request is forwarded to the backend pods Nginx-1 and Nginx-2.

For more information about DNS resolutions in Kubernetes clusters, see Introduction and configuration of the DNS service in ACK clusters.

Related information

- Introduction and configuration of the DNS service in ACK clusters
- Optimize DNS resolution for an ACK cluster
- Use NodeLocal DNSCache in an ACK cluster

6.6.2. Introduction and configuration of the DNS service in ACK clusters

In a cluster of Container Service for Kubernetes (ACK), Service names are preferably used to access Services. Therefore, a cluster-wide DNS service is required to resolve Service names to cluster IPs (IP addresses assigned to Services). CoreDNS is deployed in ACK clusters and serves as a DNS server to provide DNS resolution services for workloads deployed in ACK clusters. This topic describes how DNS resolution works in ACK clusters and how to configure CoreDNS and DNS policies in different scenarios.

Prerequisites

Before you configure the pre-installed DNS server, make sure that you have completed the following steps:

- 创建Kubernetes托管版集群
- Connect to Kubernetes clusters by using kubectl

Context

By default, a Service named kube-dns is deployed in an ACK cluster to provide DNS resolution service for the ACK cluster. You can run the following command to query information about the kube-dns Service:

```
kubectl get svc kube-dns -n kube-system
```

The following output is returned:

```
NAME    TYPE    CLUSTER-IP   EXTERNAL-IP  PORT(S)          AGE
kube-dns  ClusterIP  172.24.0.10  <none>    53/UDP,53/TCP,9153/TCP  27d
```

Two pods named coredns are deployed as the backend of the DNS resolution service for the ACK cluster. You can run the following command to query information about the two pods named coredns.

```
kubectl get deployment coredns -n kube-system
```

The following output is returned:

```
NAME    READY  UP-TO-DATE  AVAILABLE  AGE
coredns  2/2   2       2      27d
```

How DNS resolution works in ACK clusters

The startup parameters of kubelet in an ACK cluster contain `--cluster-dns=<dns-service-ip>` and `--cluster-domain=<default-local-domain>`. These parameters are used to configure the IP addresses and suffixes of the base domain name for the DNS servers in the ACK cluster.

The DNS configuration file in the pod is */etc/resolv.conf*. The file contains the following content:

```
nameserver xx.xx.0.10
search kube-system.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
```

Parameters in the DNS configuration file

Parameter	Description
nameserver	Specifies the IP addresses of the DNS servers.
search	Specifies the suffixes that are used for DNS queries. More suffixes indicate more DNS queries. In ACK clusters, suffixes are `kube-system.svc.cluster.local`, `svc.cluster.local`, and `cluster.local`. Therefore, up to eight queries (four for an IPv4 address and four for an IPv6 address) are generated for a request sent to an ACK cluster.
options	Specifies the options for the DNS configuration file. You can specify multiple key-value pairs. For example, ndots:5 specifies that if the number of dots in the domain name string is greater than 5, the domain name is a fully qualified domain name and is directly resolved. If the number of dots in the domain name string is less than 5, the domain name is appended with the suffixes specified by the search parameter before it is resolved.

Based on the preceding DNS resolution settings on the pods, queries of internal domain names (Services) and external domain names are sent to the DNS servers of an ACK cluster for DNS resolution.

Configure DNS policies for an ACK cluster in different scenarios

You can use dnsPolicy to specify different DNS policies among pods. ACK clusters support the following DNS policies:

- ClusterFirst: This policy indicates that a pod uses CoreDNS to resolve domain names from inside or outside of the ACK cluster. The */etc/resolv.conf* file contains the address of the DNS server that is provided by CoreDNS, which is kube-dns. This is the default DNS policy for workloads in an ACK cluster.
- None: This policy indicates that a pod ignores the DNS settings of the ACK cluster. You must customize the DNS settings by using the dnsConfig field.
- Default: This policy indicates that a pod inherits the DNS resolution settings from the node where the pod is deployed. In an ACK cluster, nodes are created based on Elastic Compute Service (ECS) instances. Therefore, a pod directly uses the */etc/resolv.conf* file of the ECS instance-based node where the pod is deployed. This file contains the address of a DNS server that is provided by Alibaba Cloud DNS.
- ClusterFirstWithHostNetwork: This policy indicates that a pod in the host network uses the ClusterFirst policy. By default, a pod in the host network uses the Default policy.

You can use the preceding DNS policies in different scenarios.

- **Scenario 1: Use CoreDNS provided by ACK clusters to resolve domain names**
  In this scenario, you must specify `dnsPolicy: ClusterFirst` for the DNS policy settings. The following code block is an example:

```
apiVersion: v1
kind: Pod
metadata:
 name: alpine
 namespace: default
spec:
 containers:
 - image: alpine
  command:
   - sleep
   - "10000"
  imagePullPolicy: Always
  name: alpine
 dnsPolicy: ClusterFirst
```

- **Scenario 2: Customize DNS settings for a pod**
  To customize DNS settings for a Deployment, you must specify `dnsPolicy: None` for the DNS policy settings. The following code block is an example:

```
apiVersion: v1
kind: Pod
metadata:
 name: alpine
 namespace: default
spec:
 containers:
 - image: alpine
  command:
   - sleep
   - "10000"
  imagePullPolicy: Always
  name: alpine
 dnsPolicy: None
 dnsConfig:
  nameservers: ["169.254.xx.xx"]
  searches:
  - default.svc.cluster.local
  - svc.cluster.local
  - cluster.local
  options:
  - name: ndots
   value: "2"
```

Descriptions of parameters in the dnsConfig section.

Parameter	Description
nameservers	A list of IP addresses of DNS servers for a pod. You can specify up to three IP addresses. If you specify dnsPolicy as `None` for a pod, you must specify at least one IP address. If you do not specify dnsPolicy as None for a pod, this parameter is optional. The listed DNS server IP addresses will be added to the nameserver parameter of the DNS configuration file that is generated based on the specified DNS policy. Duplicate IP addresses are removed.
searches	A list of DNS search domains for hostname lookup in a pod. This parameter is optional. The listed DNS search domains will be added to the list of base search domains that are generated based on the specified DNS policy. Duplicate domain names are removed. You can specify up to six search domains.
options	A list of optional items. Each item can contain a name (required) and a value (optional). The specified items will be added to the list of optional items that are generated based on the specified DNS policy. Duplicate items are removed.

For more information, see DNS for Services and Pods.

- **Scenario 3: Use the DNS settings of an ECS instance that is provided by Alibaba Cloud**
  If your application pods do not need to access other services deployed in the ACK cluster, you can specify `dnsPolicy: Default` for the DNS policy settings. In this scenario, the DNS resolution is performed by Alibaba Cloud DNS and CoreDNS is not required. The following code block is an example:

```
apiVersion: v1
kind: Pod
metadata:
 name: alpine
 namespace: default
spec:
 containers:
 - image: alpine
  command:
   - sleep
   - "10000"
  imagePullPolicy: Always
  name: alpine
 dnsPolicy: Default
```

- **Scenario 4: Enable pods in the host network to access services of an ACK cluster**
  If you specify hostNetwork:true for the network settings of your application pods, your application pods can directly use the host network. In this case, the default DNS policy for a pod is Default. As a result, your application pods cannot access services deployed in the ACK cluster. If you want to enable pods that run in the host network to access services deployed in the ACK cluster, you must specify `dnsPolicy: ClusterFirstWithHostNet` for the DNS policy settings. The following code block is an example:

```
apiVersion: v1
kind: Pod
metadata:
 name: alpine
 namespace: default
spec:
 hostNetwork: true
 dnsPolicy: ClusterFirstWithHostNet
 containers:
 - image: alpine
  command:
   - sleep
   - "10000"
  imagePullPolicy: Always
  name: alpine
```

Configure CoreDNS for an ACK cluster

This section describes the default configurations of CoreDNS and how to configure extended features based on CoreDNS. This applies to scenarios as described in Scenario 1: Use CoreDNS provided by ACK clusters to resolve domain names.

> **Note** If local-dns is installed in your ACK cluster, you can find a configuration file named node-local-dns in the kube-system namespace. The configuration file of local-dns is configured in the same way as that of CoreDNS.

Default configurations of CoreDNS

In the kube-system namespace, you can find a ConfigMap named coredns. For more information about how to view a ConfigMap, see View a ConfigMap. The following code block is an example for the Corefile field of the coredns ConfigMap. CoreDNS provides services based on the plug-ins specified in the Corefile field.

```
Corefile: |
 .:53 {
    errors
    health {
      lameduck 5s
    }
    ready
    kubernetes cluster.local in-addr.arpa ip6.arpa {
     pods insecure
     upstream
     fallthrough in-addr.arpa ip6.arpa
     ttl 30
    }
    prometheus :9153
    forward . /etc/resolv.conf
    cache 30
    loop
    reload
    loadbalance
 }
```

Descriptions of default add-ons

Parameter	Description
errors	Errors are logged to stdout.
health	The health check report of CoreDNS. The default listening port is 8080. This parameter is used to evaluate the health status of CoreDNS. You can visit `http://localhost:8080/health` to view the health check report of CoreDNS.
ready	The health check report of the CoreDNS plug-ins. The default listening port number is 8181. This parameter is used to evaluate the readiness of the CoreDNS plug-ins. You can visit `http://localhost:8181/ready` to view the readiness of the CoreDNS plug-ins. After all plug-ins are in the running state, 200 is returned for the readiness of CoreDNS plug-ins.
kubernetes	The CoreDNS kubernetes plug-in that is used to provide DNS resolution for Services inside an ACK cluster.
prometheus	Metrics of CoreDNS. You can visit `http://localhost:9153/metrics` to view metrics of CoreDNS in Prometheus format.
forward or proxy	DNS queries are forwarded to the predefined DNS server. By default, DNS queries of domain names that are not within the cluster domain of Kubernetes are forwarded to the predefined resolver (*/etc/resolv.conf*). The default configurations are based on the */etc/resolv.conf* file on the host.

Parameter	Description
cache	DNS cache.
loop	Loop detection. If a loop is detected, CoreDNS is suspended.
reload	Allows automatic reload of a changed Corefile. After you edit the ConfigMap configuration, wait two minutes for the changes to take effect.
loadbalance	A round-robin DNS loadbalancer that randomizes the order of A, AAAA, and MX records in the answer.

Configure extended features based on CoreDNS

You can configure extended features based on CoreDNS in the following scenarios:

- **Scenario 1: Enable Log Service**
  To log each DNS resolution performed by CoreDNS, enable the log plug-in by adding log to Corefile. The following code block is an example:

```
Corefile: |
  .:53 {
    errors
    log
    health {
      lameduck 5s
    }
    ready
    kubernetes cluster.local in-addr.arpa ip6.arpa {
      pods insecure
      upstream
      fallthrough in-addr.arpa ip6.arpa
      ttl 30
    }
    prometheus :9153
    forward . /etc/resolv.conf
    cache 30
    loop
    reload
    loadbalance
  }
```

- **Scenario 2: Customize DNS servers for specified domain names**
  If domain names with a suffix of example.com need to be resolved by a user-defined DNS server (for example, DNS server 10.10.0.10), you must add a custom resolution setting for the domain names. The following code block is an example:

```
example.com:53 {
  errors
  cache 30
  forward . 10.10.0.10
}
```

  The following code block shows all configurations:

```
Corefile: |
 .:53 {
    errors
    health {
      lameduck 5s
    }
    ready
    kubernetes cluster.local in-addr.arpa ip6.arpa {
     pods insecure
     upstream
     fallthrough in-addr.arpa ip6.arpa
     ttl 30
    }
    prometheus :9153
    forward . /etc/resolv.conf
    cache 30
    loop
    reload
    loadbalance
 }
 example.com:53 {
    errors
    cache 30
    forward . 10.10.0.10
 }
```

- **Scenario 3: Customize DNS servers for external domain names**
  If the domain names that need to be resolved by user-defined DNS servers do not contain the same suffix, you can use user-defined DNS servers to perform DNS resolution for all external domain names. In this scenario, you must forward the domain names that cannot be resolved by the on-premises DNS servers to Alibaba Cloud DNS. Do not modify the */etc/resolv.conf* files on ECS instances of the ACK cluster. Assume that the IP addresses of the user-defined DNS servers are 10.10.0.10 and 10.10.0.20, you can modify the forward parameter. The following code block is an example:

```
Corefile: |
 .:53 {
    errors
    health {
      lameduck 5s
    }
    ready
    kubernetes cluster.local in-addr.arpa ip6.arpa {
     pods insecure
     upstream
     fallthrough in-addr.arpa ip6.arpa
     ttl 30
    }
    prometheus :9153
    forward . 10.10.0.10 10.10.0.20
    cache 30
    loop
    reload
    loadbalance
 }
```

- **Scenario 4: Customize hosts for specified domain names**

You can configure the hosts plug-in if you want to customize hosts for specified domain names, for example, you may need to point www.example.com to 127.0.0.1. The following code block is an example:

```
Corefile: |
 .:53 {
    errors
    health {
      lameduck 5s
    }
    ready
    hosts {
     127.0.0.1 www.example.com
     fallthrough
    }
    kubernetes cluster.local in-addr.arpa ip6.arpa {
     pods insecure
     upstream
     fallthrough in-addr.arpa ip6.arpa
     ttl 30
    }
    prometheus :9153
    forward . /etc/resolv.conf
    cache 30
    loop
    reload
    loadbalance
 }
```

Notice You must specify fallthrough. Otherwise, domain names other than the specified one may fail to be resolved.

- **Scenario 5: Enable external access to services in an ACK cluster**
  If you want to enable a process on an ECS instance to access services in the ACK cluster where the ECS instance is deployed, you can add the ClusterIP of kube-dns in the ACK cluster to the nameserver parameter of the */etc/resolv.conf* file on the ECS instance. Do not modify other settings in the */etc/resolv.conf* file.
  In an internal network, you can use an internal Server Load Balancer (SLB) instance to allow internal access to services in the ACK cluster. Then, log on to the Alibaba Cloud DNS console and add an A record that points to the private IP address of the SLB instance.
- **Scenario 6: Use a domain name to allow access to your service in an ACK cluster or enable CNAME resolution for an ACK cluster**
  You can use foo.example.com to allow all access to your service in an ACK cluster from the Internet, internal networks, and inside an ACK cluster. The following section describes how to enable this feature:
  - Your service foo.default.svc.cluster.local is exposed to external access through an public-facing SLB instance. The domain name foo.example.com is resolved to the IP address of the public-facing SLB instance.
  - Your service foo.default.svc.cluster.local is exposed to internal access through an public-facing SLB instance. Log on to the Alibaba Cloud DNS console to point foo.example.com to the IP address of the internal SLB instance in the virtual private cloud (VPC) where the ACK cluster is deployed. For more information about the procedure, see the preceding section Scenario 4: Customize hosts for specified domain names.

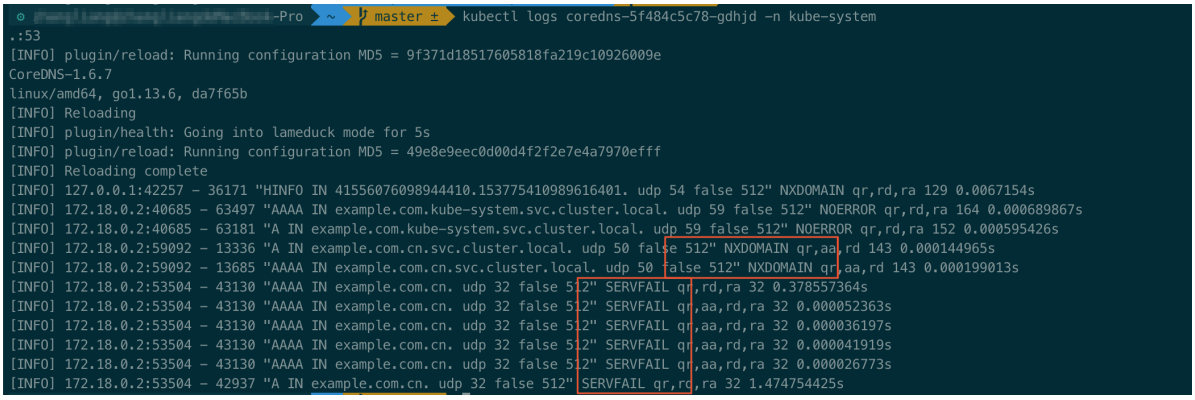
- Inside the ACK cluster, you can use the Rewrite plug-in to add a Canonical Name (CNAME) record to point foo.example.com to foo.default.svc.cluster.local. The following code block is an example:

```
Corefile: |
 .:53 {
    errors
    health {
      lameduck 5s
    }
    ready
    rewrite stop {
     name regex foo.example.com foo.default.svc.cluster.local
     answer name foo.default.svc.cluster.local foo.example.com
    }
    kubernetes cluster.local in-addr.arpa ip6.arpa {
     pods insecure
     upstream
     fallthrough in-addr.arpa ip6.arpa
     ttl 30
    }
    prometheus :9153
    forward . /etc/resolv.conf
    cache 30
    loop
    reload
    loadbalance
 }
```

- **Scenario 7: Monitor failed DNS resolutions of CoreDNS**
  You can query the CoreDNS log to troubleshoot failed DNS resolutions of CoreDNS. For more information, see the preceding section Scenario 1: Enable Log Service. In this case, you must enable autopath for CoreDNS. For more information about autopath, see Optimize DNS resolution for an ACK cluster. The following code block is an example:

```
Corefile: |
 .:53 {
    errors
    health
    ready
    log
    kubernetes cluster.local in-addr.arpa ip6.arpa {
     pods verified
     fallthrough in-addr.arpa ip6.arpa
    }
    autopath @kubernetes
    prometheus :9153
    forward . /etc/resolv.conf
    cache 30
    loop
    reload
    loadbalance
 }
```

After the template is implemented, run the `kubectl get pods -n kube-system | grep coredns` command to query information about the CoreDNS pods. Then, run the `kubectl logs coredns-{pod id} -n kube-system` command to query the log of each CoreDNS pod. If response codes of the NXDOMAIN or SERVFAIL type appear in the log of CoreDNS, it indicates failed DNS resolutions of CoreDNS.



6.6.3. Optimize DNS resolution for an ACK cluster

CoreDNS is a Domain Name System (DNS) resolver for Kubernetes clusters. CoreDNS resolves custom domain names from inside or outside Kubernetes clusters. CoreDNS contains a variety of add-ons. These add-ons allow you to configure custom DNS settings and customize hosts, Canonical Name (CNAME) records, and rewrite rules for Kubernetes clusters. Container Service for Kubernetes (ACK) clusters require high queries per second (QPS) for DNS resolution. DNS queries may fail if the QPS cannot match the heavy load. This topic describes how to optimize DNS resolution for an ACK cluster.

Prerequisites

- 创建Kubernetes托管版集群
- Connect to Kubernetes clusters by using kubectl

Context

For more information about CoreDNS, see CoreDNS.

Adjust the number of CoreDNS pods

Setting a proper ratio of CoreDNS pods to nodes in a Kubernetes cluster improves the performance of service discovery for the cluster. We recommend that you set the ratio to 1:8.

- If you do not want to expand your cluster on a large scale, you can run the following command to change the number of pods:

  ```
  kubectl scale --replicas={target} deployment/coredns -n kube-system
  ```

  > **Note** Replace `{target}` with the required value.

- If you want to expand your cluster on a large scale, you can use the following YAML template to create a Deployment. Then, you can use cluster-proportional-autoscaler to dynamically scale the number of pods. We recommend that you do not use Horizontal Pod Autoscaler (HPA) to scale the number of pods when the QPS, CPU utilization, or memory usage of pods reaches the threshold. HPA cannot meet the expected requirements.

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: dns-autoscaler
 namespace: kube-system
 labels:
  k8s-app: dns-autoscaler
spec:
 selector:
  matchLabels:
   k8s-app: dns-autoscaler
 template:
  metadata:
   labels:
    k8s-app: dns-autoscaler
  spec:
   serviceAccountName: admin
   containers:
   - name: autoscaler
    image: registry.cn-hangzhou.aliyuncs.com/acs/cluster-proportional-autoscaler:1.8.4
    resources:
     requests:
      cpu: "200m"
      memory: "150Mi"
    command:
    - /cluster-proportional-autoscaler
    - --namespace=kube-system
    - --configmap=dns-autoscaler
    - --nodelabels=type!=virtual-kubelet
    - --target=Deployment/coredns
    - --default-params={"linear":{"coresPerReplica":64,"nodesPerReplica":8,"min":2,"max":100,"preventSingle
PointFailure":true}}
    - --logtostderr=true
    - --v=9
```

Note In the preceding linear scaling policy, the number of replicas for CoreDNS is calculated based on the following formula: Replicas = Max (Ceil (Cores × 1/coresPerReplica), Ceil (Nodes × 1/nodesPerReplica)). The number of CoreDNS pod replicas is subject to the values of `max` and `min` in the linear scaling policy.
The following code block shows the parameters of the linear scaling policy:

```
{
    "coresPerReplica": 64,
    "nodesPerReplica": 8,
    "min": 2,
    "max": 100,
    "preventSinglePointFailure": true
}
```

Use NodeLocal DNSCache in a Kubernetes cluster

ACK allows you to deploy NodeLocal DNSCache to improve the stability and performance of service discovery. NodeLocal DNSCache is implemented as a DaemonSet and runs a DNS caching agent on cluster nodes to improve the DNS performance.

For more information about NodeLocal DNSCache and how to deploy NodeLocal DNSCache in ACK clusters, see Use NodeLocal DNSCache in an ACK cluster

Related information

- Overview
- Introduction and configuration of the DNS service in ACK clusters
- Use NodeLocal DNSCache in an ACK cluster

6.6.4. Use NodeLocal DNSCache in an ACK cluster

Container Service for Kubernetes (ACK) allows you to deploy NodeLocal DNSCache to improve the stability and performance of service discovery. NodeLocal DNSCache is implemented as a DaemonSet and runs a DNS caching agent on cluster nodes to improve cluster DNS performance. This topic describes how to deploy and configure NodeLocal DNSCache for an application in an ACK cluster.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- A kubectl client is connected to the cluster. For more information, see Connect to Kubernetes clusters by using kubectl.

Limits

- NodeLocal DNSCache does not support pods that run in serverless Kubernetes (ASK) clusters or on elastic container instances in managed or dedicated Kubernetes clusters.
- The Terway version must be v1.0.10.301 or later.
- To install NodeLocal DNSCache, you can go to the Add-ons page or install ack-node-local-dns on the App Catalog page.
- NodeLocal DNSCache serves as a transparent cache proxy for CoreDNS and does not provide features such as Hosts or Rewrite. If you want to enable these features, modify CoreDNS configurations.
- If PrivateZone is used in the cluster, you must modify CoreDNS configurations before you enable PrivateZone. For more information, see Compatibility with PrivateZone.

Introduction

ACK NodeLocal DNSCache is a local DNS cache solution developed based on the open source NodeLocal DNSCache project. ACK NodeLocal DNSCache is the ack-node-local-dns chart in Helm. This solution consists of a DNS cache that runs as a DaemonSet and an admission controller that runs as a Deployment to dynamically inject DNSConfig.

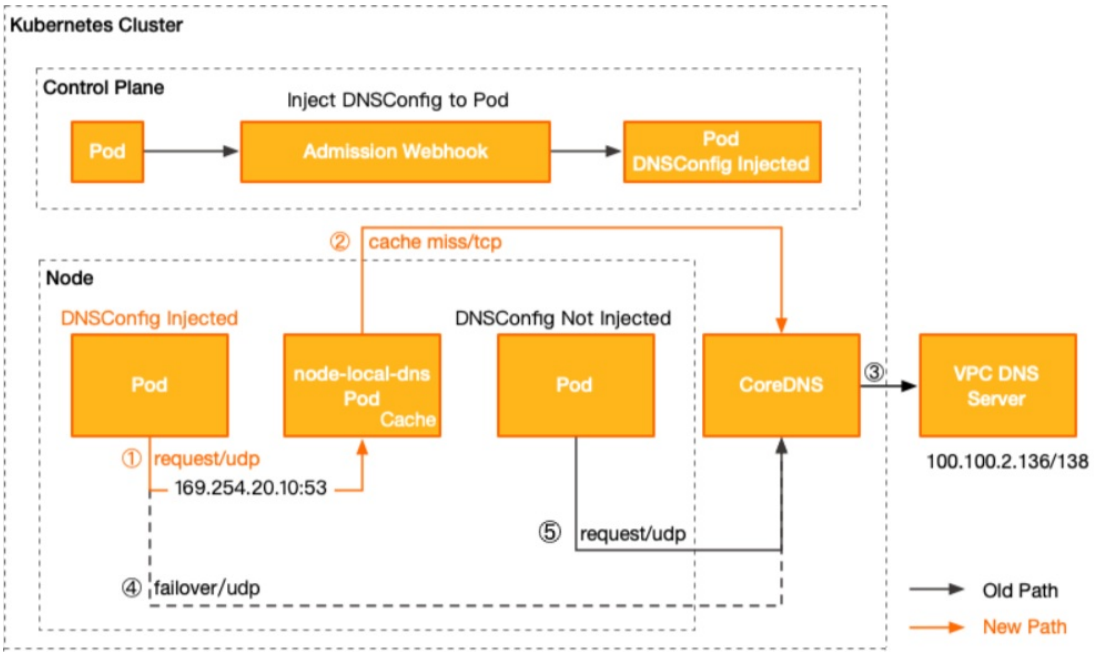
- The admission controller intercepts pod creation requests based on admission webhooks and dynamically injects cached DNSConfig to pod configurations.

  > **Note** If you do not enable the admission controller to automatically inject DNSConfig, you must manually add DNS settings to pod configurations. For more information, see Method 2: Manually specify DNSConfig in Configure NodeLocal DNSCache for an application.

- The DaemonSet that runs a DNS cache on each node can create a virtual network interface. By default, the virtual network interface listens for DNS queries that are sent to the IP address 169.254.20.10. To change the IP address, Submit a ticket. DNS queries that are generated in pods are proxied by the DaemonSet based on pod DNSConfig and node network settings.

  > **Notice** The DaemonSets that run DNS caches are built on CoreDNS and provide only proxy and cache services. Do not enable other features such as Hosts and Rewrite on the DaemonSets.

How it works



No.	Description
1	By default, a pod with the local DNSConfig injected uses NodeLocal DNSCache to listen for DNS queries that are sent to the IP address 169.254.20.10 on the node.
2	If NodeLocal DNSCache does not find a cache hit for the DNS query, the kube-dns Service is used to request CoreDNS for DNS resolution.
3	CoreDNS uses the DNS server deployed in the virtual private cloud (VPC) to resolve non-cluster domain names.
4	When the pod with the local DNSConfig injected fails to connect to NodeLocal DNSCache, the pod uses the kube-dns Service to connect to CoreDNS for DNS resolution.
5	A pod without the local DNSConfig injected uses the kube-dns Service to connect to CoreDNS for DNS resolution.

Install NodeLocal DNSCache

You can install NodeLocal DNSCache on the Add-ons page or the App Catalog page. We recommend that you install NodeLocal DNSCache on the Add-ons page.

Install NodeLocal DNSCache on the Add-ons page

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and choose **More > Manage System Components** in the **Actions** column.
4. On the **Add-ons** page, click the **Networking** tab and find ACK NodeLocal DNSCache.
5. Click **Install**. In the dialog box that appears, click **OK**.

Install NodeLocal DNSCache on the App Catalog page

1. Log on to the ACK console.

2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the App Catalog page, search for ack-node-local-dns. In the search result, click **ack-node-local-dns**.
4. On the **Description** tab, read the introduction and parameter settings about **ack-node-local-dns**. Then, click the **Parameters** tab to set the parameters.

   The following table describes the parameters.

Parameter	Description	How to obtain the value
upstream_ip	The cluster IP of the kube-dns Service in the kube-system namespace. NodeLocal DNSCache uses the kube-dns Service to communicate with CoreDNS and resolve cluster-local domain names.	You can modify the parameter value to specify other upstream DNS servers.
clusterDomain	The domain name of the cluster.	Query the --cluster-domain parameter of the kubelet process on a cluster node. The default value of the parameter is cluster.local.

5. In the **Deploy** panel on the right side of the page, select the cluster and namespace, specify a release name, and then click **Create**.

Configure NodeLocal DNSCache for an application

> **Note** By default, NodeLocal DNSCache is not installed on master nodes. If you want to install NodeLocal DNSCache on master nodes and taints are configured on master nodes, you must modify taints and tolerations for the node-local-dns DaemonSet in the kube-system namespace.

To forward DNS requests that are previously directed to CoreDNS to the DaemonSet that runs a DNS cache, you must set the nameservers parameter of pod configurations to 169.254.20.10 and the IP address of kube-dns. To do this, use one of the following methods:

- Method 1: Use the admission controller to automatically inject DNSConfig when pods are created. We recommend that you use this method.
- Method 2: Manually specify DNSConfig when pods are created.
- Method 3: Modify kubelet parameters and restart kubelet. We recommend that you do not use this method because your business may be interrupted.

Method 1: Automatically inject DNSConfig

You can use an admission controller to automatically inject DNSConfig to newly created pods. This way, you do not need to manually configure the pod configuration file. By default, the application listens for requests to create pods from namespaces that have the `node-local-dns-injection=enabled` label. You can use the following command to add this label to a namespace.

```
kubectl label namespace default node-local-dns-injection=enabled
```

> **Note**
>
> - The preceding command enables automatic DNSConfig injection only for the default namespace. To enable DNSConfig auto injection for other namespaces, replace `default` with the namespace based on your requirements.
> - If automatic DNSConfig injection is enabled for a namespace but you do not want to automatically inject DNSConfig to a pod, you can add the `node-local-dns-injection=disabled` label to the pod template.

After automatic DNSConfig injection is enabled, the following parameters are added to new pods. To ensure the high availability of DNS services, the cluster IP address of kube-dns is added to the nameservers parameter as a backup DNS server.

```
dnsConfig:
  nameservers:
  - 169.254.20.10
  - 172.21.0.10
  options:
  - name: ndots
    value: "3"
  - name: attempts
    value: "2"
  - name: timeout
    value: "1"
  searches:
  - default.svc.cluster.local
  - svc.cluster.local
  - cluster.local
 dnsPolicy: None
```

To enable automatic DNSConfig injection, the following conditions must be met. If DNSConfig is not automatically injected, check whether the conditions are met:

- New pods do not belong to the kube-system or kube-public namespace.
- The namespace to which new pods belong has the `node-local-dns-injection=enabled` label.
- The namespace to which new pods belong does not have labels that are related to Elastic Container Instance-based pods, such as `virtual-node-affinity-injection`, `eci`, and `alibabacloud.com/eci`.
- New pods do not have labels that are related to elastic container instances, such as `eci` and `alibabacloud.com/eci`, or the `node-local-dns-injection=disabled` label.
- New pods have the `hostNetwork` network and the `ClusterFirstWithHostNet` DNSPolicy, or the pods do not have the `hostNetwork` network but have the `ClusterFirst` DNSPolicy.

Method 2: Manually specify DNSConfig

If you do not want to use admission webhooks to automatically inject DNSConfig, you can modify pod configurations to manually specify DNSConfig.

```
apiVersion: v1
kind: Pod
metadata:
 name: alpine
 namespace: default
spec:
 containers:
 - image: alpine
  command:
   - sleep
   - "10000"
  imagePullPolicy: Always
  name: alpine
 dnsPolicy: None
 dnsConfig:
  nameservers: ["169.254.20.10","172.21.0.10"]
  searches:
  - default.svc.cluster.local
  - svc.cluster.local
  - cluster.local
  options:
  - name: ndots
   value: "3"
  - name: attempts
   value: "2"
  - name: timeout
   value: "1"
```

- dnsPolicy: Set the value to `None`.
- nameservers: Set the value to 169.254.20.10 and the cluster IP address of kube-dns.
- searches: Set the DNS search domains. Make sure that internal domains can be resolved.
- ndots: Default value: 5. You can improve resolution efficiency by setting this parameter to a smaller value. For more information, see resolv.conf.

Method 3: Configure kubelet startup parameters

The kubelet uses the --cluster-dns and --cluster-domain parameters to control pod DNSConfig. In the */etc/systemd/system/kubelet.service.d/10-kubeadm.conf* file, add the --cluster-dns parameter and set the value to the local IP address 169.254.20.10. Then, run the `systemctl daemon-reload` and `systemctl restart kubelet` commands for the changes to take effect.

```
--cluster-dns=169.254.20.10 --cluster-dns=<kube-dns ip> --cluster-domain=<search domain>
```

- cluster-dns: specifies the DNS servers that are used in the pod configuration. By default, only the IP address of `` `kube-dns` `` is specified. You must add the local IP address 169.254.20.10.
- cluster-domain: specifies the DNS search domains that are used in the pod configuration. You can use the existing domain. In most cases, the existing domain is `` `cluster.local` ``.

Example on how to enable NodeLocal DNSCache for an application

The following example shows how to enable NodeLocal DNSCache for a Deployment that is created in the default namespace.

1. Run the following command to add a label to the namespace to which the Deployment belongs. In this example, the Deployment is created in the default namespace.

```
kubectl label namespace default node-local-dns-injection=enabled
```

Notice The admission controller ignores applications in the kube-system and kube-public namespaces. Do not configure to automatically inject dnsConfig to applications in the two namespaces.

2. Deploy a sample application in the default namespace.
   i. Use the following YAML template to create a sample application named ubuntu-deployment:

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
 name: ubuntu
 labels:
  app: ubuntu
spec:
 replicas: 2
 selector:
  matchLabels:
   app: ubuntu
 template:
  metadata:
   labels:
    app: ubuntu
  spec:
   containers:
   - name: ubuntu
    image: ubuntu
    command: ["sh", "-c"]
    args: ["sleep 100000"]
```

   ii. Run the following command to deploy the application in the cluster:

```
kubectl apply -f ubuntu-deployment.yaml
```

   Expected output:

```
deployment.apps/ubuntu created
```

   iii. Run the following command to view the application information:

```
kubectl get deployment ubuntu
```

   Expected output:

```
NAME   READY  UP-TO-DATE  AVAILABLE  AGE
ubuntu  2/2   2      2      7s
```

3. Check whether dnsConfig is injected.

i. Run the following command to view the pods that are provisioned for the application:

```
kubectl get pods
```

Expected output:

```
NAME          READY  STATUS   RESTARTS  AGE
ubuntu-766448f68c-mj8qk  1/1   Running  0     4m39s
ubuntu-766448f68c-wf5hw  1/1   Running  0     4m39s
```

ii. Run the following command to check whether NodeLocal DNSCache is enabled in dnsConfig of a pod:

```
kubectl get pod ubuntu-766448f68c-mj8qk -o=jsonpath='{.spec.dnsConfig}'
```

Expected output:

```
map[nameservers:[169.254.20.10 172.21.0.10] options:[map[name:ndots value:5]] searches:[default.svc.cluster.local svc.cluster.local cluster.local]]
```

The preceding output indicates that NodeLocal DNSCache is enabled for the application. After NodeLocal DNSCache is enabled for an application, the following parameters are added to the pods that are provisioned for the application. To ensure high availability of DNS services, the cluster IP address 172.21.0.10 of the kube-dns Service is specified as the IP address of the backup DNS server in the nameservers parameter.

```
dnsConfig:
 nameservers:
 - 169.254.20.10
 - 172.21.0.10
 options:
 - name: ndots
  value: "3"
 - name: attempts
  value: "2"
 - name: timeout
  value: "1"
 searches:
 - default.svc.cluster.local
 - svc.cluster.local
 - cluster.local
dnsPolicy: None
```

Upgrade NodeLocal DNSCache

If you have installed NodeLocal DNSCache on the Add-ons page, you must upgrade NodeLocal DNSCache on the Add-ons page. If you have installed NodeLocal DNSCache on the App Catalog page, you must uninstall the current version of NodeLocal DNSCache on the Helm chart page and install the latest version.

Upgrade NodeLocal DNSCache on the Add-ons page

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. On the **Add-ons** page, find NodeLocal DNSCache and click **Upgrade**. In the dialog box that appears,

click **OK**.

> **Note** If you modified taints and tolerations for the node-local-dns DaemonSet, the modifications are overwritten during the upgrade process. You must configure taints and tolerations again after the upgrade is completed.

Upgrade NodeLocal DNSCache on the App Catalog page

If you have installed NodeLocal DNSCache on the App Catalog page, you must uninstall the current version of NodeLocal DNSCache and install the latest version. For more information, see Uninstall NodeLocal DNSCache and Install NodeLocal DNSCache.

Uninstall NodeLocal DNSCache

If you have installed NodeLocal DNSCache on the Add-ons page, you must uninstall NodeLocal DNSCache on the Add-ons page. If you have installed NodeLocal DNSCache on the App Catalog page, you must uninstall NodeLocal DNSCache on the Helm chart page.

Uninstall NodeLocal DNSCache on the Add-ons page

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. On the **Add-ons** page, find NodeLocal DNSCache and click **Uninstall**. In the dialog box that appears, click **OK**.

> **Note** After NodeLocal DNSCache is uninstalled, all DNS queries are sent to CoreDNS. We recommend that you add replicas for CoreDNS before you uninstall NodeLocal DNSCache.

Uninstall NodeLocal DNSCache on the App Catalog page

1. Check the parameter settings.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
   iv. In the left-side navigation pane of the details page, choose **Applications > Helm**.
   v. On the **Helm** page, find and click **ack-node-local-dns-default**.

vi. Click the **Parameters** tab and check the parameter settings.

- If the local_dns_ip parameter is set to the cluster IP address of kube-dns, proceed to Step 2 to uninstall NodeLocal DNSCache.
- If the high_availability parameter is set to enabled and the local_dns_ip parameter is set to 169.254.20.10, proceed to Step 2 to uninstall NodeLocal DNSCache.
- If the parameters do not meet the preceding conditions, perform the following steps to uninstall NodeLocal DNSCache:
  a. Run the `kubectl get ns -o yaml` command to find the namespaces that have the node-local-dns-injection=enabled label. Remove the label from the namespaces.
  b. Run the `kubectl get pod -o yaml` command to find the pods to which DNSConfig is injected. Delete the pods and create new pods.
  c. Perform Step 2 to uninstall NodeLocal DNSCache.

2. Uninstall NodeLocal DNSCache.
   i. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
   iv. In the left-side navigation pane of the details page, choose **Applications > Helm**.
   v. On the **Helm** page, select ack-node-local-dns-default and click **Delete** in the **Actions** column. In the dialog box that appears, click **OK**.

Compatibility with PrivateZone

By default, NodeLocal DNSCache uses the TCP protocol to communicate with CoreDNS. CoreDNS communicates with the upstream servers based on the protocol used by the source of DNS queries. If PrivateZone is used in your cluster, DNS queries processed by NodeLocal DNSCache are sent to PrivateZone by using the TCP protocol. PrivateZone does not support the TCP protocol in some regions. Therefore, errors may occur when DNS queries are processed by using PrivateZone.

Related information

- Overview
- Introduction and configuration of the DNS service in ACK clusters
- Optimize DNS resolution for an ACK cluster

6.7. FAQ about network management

This topic provides answers to some frequently asked questions about container networks, Services, and Ingresses.

FAQ about container networks

- How do I solve the issue that Flannel becomes incompatible with clusters of Kubernetes 1.16 or later after I manually upgrade Flannel?
- How do I resolve the issue that a pod is not immediately ready for communication after it is started?
- How do I enable a pod to access the Service that is exposed on it?
- Which network plug-in should I choose for an ACK cluster, Terway or Flannel?
- How do I plan the network of a cluster?
- Can I use hostPorts to create port mappings in an ACK cluster?
- Can I configure multiple route tables for the VPC where my cluster is deployed?

- How do I check the network type and vSwitches of the cluster?
- How do I check the cloud resources used in an ACK cluster?
- Network errors of pods in the cluster
- How do I obtain the public IP address of an application in the cluster?
- Network errors that occur when the Terway network plug-in is used in the exclusive ENI mode
- How do I troubleshoot cluster access issues?
- The number of IP addresses provided by the vSwitch is insufficient when the Terway network plug-in is used
- The cluster cannot connect to the public IP address of the SLB instance that is associated with the LoadBalancer Service

Service FAQ

FAQ about Server Load Balancer (SLB)

- Why are no events collected during the synchronization between a Service and an SLB instance?
- How do I handle an SLB instance that remains in the Pending state?
- What do I do if the vServer groups of an SLB instance are not updated?
- What do I do if the annotations of a Service do not take effect?
- Why is the configuration of an SLB instance modified?
- Why does the cluster fail to access the IP address of the SLB instance?
- What do I do if I accidentally delete an SLB instance?
- If I delete a Service, is the SLB instance associated with the Service automatically deleted?
- How do I rename an SLB instance when the CCM version is V1.9.3.10 or earlier?
- How does CCM calculate node weights in Local mode?

FAQ about Cloud Controller Manager (CCM) upgrades

- How do I troubleshoot failures to upgrade CCM?

FAQ about using existing SLB instances

- Why does the system fail to use an existing SLB instance for more than one Services?
- Why is no listener created when I reuse an existing SLB instance?

Other FAQ

- How Do I Perform Session Persistence for a Kubernetes Service?

Ingress FAQ

- Which Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol versions are supported by Ingresses?
- Do Ingresses pass Layer 7 request headers?
- Can ingress-nginx forward requests to HTTPS Services at the backend?
- Do Ingresses pass client IP addresses at Layer 7?
- Does nginx-ingress-controller support HTTP Strict Transport Security (HSTS)?
- Which rewrite rules are supported by ingress-nginx?
- How do I fix the issue that Log Service cannot parse logs as expected after ingress-nginx-controller is upgraded?
- How do I configure the internal-facing SLB instance for the NGINX Ingress controller?
- What are the system updates after I upgrade the NGINX Ingress controller on the Add-ons page in the

console?

7.Application

7.1. Workloads

7.1.1. Create a stateless application by using a Deployment

You can deploy a stateless application by using a Deployment. A Deployment can be created by using an image, an orchestration template, or kubectl commands. Container Service for Kubernetes (ACK) allows you to use Secrets to pull images by using a web interface. This topic describes how to create a stateless NGINX application by using an image, an orchestration template, and kubectl commands.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- Your machine is connected to the ACK cluster by using kubectl. For more information, see Use kubectl to connect to an ACK cluster.
- A private image repository is created and your image is uploaded to the repository. In this topic, an image repository from Container Registry is used. For more information, see Manage images.

Create a Deployment from an image

Step 1: Configure basic settings

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. In the upper-right corner of the **Deployments** page, click **Create from Image**.
6. On the **Basic Information** wizard page, configure the basic settings of the application.

   Before you configure the Deployment, select a namespace in the upper part of the page. In this example, the **default** namespace is selected.

Parameter	Description
Name	The name of the application.
Replicas	The number of pods that are provisioned for the application. Default value: 2.
Type	The type of application. You can select **Deployments**, **StatefulSets**, **Jobs**, **CronJobs**, or **DaemonSets**.
Label	Add a label to the application. The label is used to identify the application.
Annotations	Add an annotation to the application.
Synchronize Timezone	Specify whether to synchronize the time zone between nodes and containers.

> **Note** In this example, **Deployment** is selected as the workload type.

7. Click **Next**.

   Proceed to the **Container** wizard page.

Step 2: Configure containers

On the **Container** wizard page, configure the following container settings: images, resources, ports, environment variables, health checks, lifecycle configurations, volumes, and Log Service configurations.

> **Note** On the **Container** wizard page, Click **Add Container** to add containers to the pods that run the application.

1. In the **General** section, configure the basic settings of the container.

Parameter	Description
Image Name	○ You can click **Select Image**. In the dialog box that appears, select an image and click **OK**. In this example, an NGINX image is selected. On the **Search** tab, select **Docker Images** from the drop-down list, enter *NGINX* into the search box, and then click Search. ■ Images from Container Registry: On the **Alibaba Cloud Container Registry** tab, you can select an image from Container Registry. You must select the region and the Container Registry instance to which the image belongs. For more information about Container Registry, see What is Container Registry?. **Note** On the **Alibaba Cloud Container Registry** tab, you can search for images by name. ■ Official Docker images: On the **Docker Official Images** tab, you can select a Docker image. ■ Favorite images: On the **Favorite Images** tab, you can select a Docker image that you have added to your favorite list. ■ Search for images: On the **Search** tab, you can select Alibaba Cloud Images from the drop-down list and specify a region to search for an image in Container Registry. You can also select Docker Images from the drop-down list and search for a Docker image. ○ You can also enter the address of a private registry. The registry address must be in the following format: `domainname/namespace/imagename:tag`.

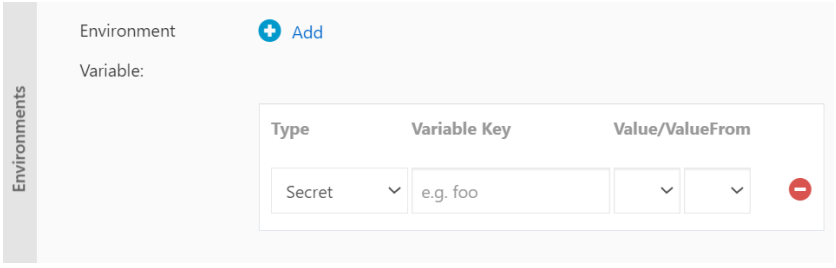
Parameter	Description
Image Version	○ Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used. ○ You can select the following image pull policies: ■ **ifNotPresent**: If the image that you want to pull is found on your on-premises machine, the image on your on-premises machine is used. Otherwise, ACK pulls the image from the corresponding repository. ■ **Always**: ACK pulls the image from Container Registry each time the application is deployed or scaled out. ■ **Never**: ACK uses only images on your on-premises machine. **Note** If you select **Image Pull Policy**, no image pull policy is applied. ○ To pull the image without a password, click **Set Image Pull Secret** to set a Secret that is used to pull the image. For more information, see Use aliyun-acr-credential-helper to pull images without a password.
Resource Limit	You can specify upper limits for the CPU, memory, and ephemeral storage resources that the container can consume. This prevents the container from occupying an excessive amount of resources.
Required Resources	The amount of CPU and memory resources that are reserved for this application. These resources are exclusive to the container. This prevents the application from becoming unavailable if other services or processes compete for computing resources.
Container Start Parameter	○ stdin: Pass stdin to the container. ○ tty: Stdin is a TeleTYpewriter (TTY).
Privileged Container	○ If you select Privileged Container, privileged=true is set for the container and the privilege mode is enabled. ○ If you do not select Privileged Container, privileged=false is set for the container and the privilege mode is disabled.
Init Container	If you select Init Container, an init container is created. An init container provides tools for pod management. For more information, see init containers.

2. (Optional)In the **Ports** section, click **Add** to configure ports for the container.

Parameter	Description
Name	Enter a name for the port.
Container Port	Specify the container port that you want to expose. The port number must be in the range of 1 to 65535.
Protocol	Valid values: TCP and UDP.

3. (Optional)In the **Environments** section, click **Add** to set environment variables.

   You can configure environment variables for pods in key-value pairs. Environment variables are used to apply pod configurations to containers. For more information, see Pod variables.

Parameter	Description
Type	Select the type of environment variable. Valid values: ○ **Custom** ○ **ConfigMaps** ○ **Secrets** ○ **Value/ValueFrom** ○ **ResourceFieldRef** If you select ConfigMaps or Secrets, you can pass all data in the selected ConfigMap or Secret to the container environment variables. In this example, Secrets is selected. Select **Secrets** from the Type drop-down list and select a Secret from the Value/ValueFrom drop-down list. By default, all data in the selected Secret is passed to the environment variable.  In this case, the *YAML* file that is used to deploy the application contains the settings that reference all data in the selected Secret. 
Variable Key	Specify the name of the environment variable.
Value/ValueFrom	Specify a reference that is used to define the environment variable.

4. (Optional)In the **Health Check** section, you can enable **liveness**, **readiness**, and **startup** probes based on your business requirements.

   For more information, see Configure Liveness, Readiness and Startup Probes.

Parameter	Request type	Description
	HTTP	Sends an HTTP GET request to the container. You can set the following parameters: ○ Protocol: Select HTTP or HTTPS. ○ Path: the requested path on the server. ○ Port: the name or the number of the container port that you want to open. The port number must be in the range of 1 to 65535. ○ HTTP Header: the custom headers in the HTTP request. Duplicate headers are allowed. Key-value pairs are supported. ○ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time (in seconds) that the system must wait before it can send a probe to the container after the container is started. Default value: 3. ○ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ○ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ○ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ○ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

Parameter	Request type	Description
○ Liveness: Liveness probes are used to determine when to restart the container. ○ Readiness: Readiness probes are used to determine whether the container is ready to accept traffic. ○ Startup: Startup probes are used to determine when to start the container. Note Startup probes are supported in Kubernetes 1.18 and later.	TCP	Sends a TCP socket to the container. kubelet attempts to open the socket on the specified port. If the connection can be established, the container is considered healthy. Otherwise, the container is considered unhealthy. You can set the following parameters: ○ Port: the name or the number of the container port that you want to open. The port number must be in the range of 1 to 65535. ○ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time (in seconds) that the system must wait before it can send a probe to the container after the container is started. Default value: 15. ○ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ○ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ○ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ○ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

Parameter	Request type	Description
	Command	Runs a probe command in the container to check the health status of the container. You can set the following parameters: ○ Command: the probe command that is run to check the health status of the container. ○ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time (in seconds) that the system must wait before it can send a probe to the container after the container is started. Default value: 5. ○ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ○ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ○ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ○ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

5. (Optional)In the **Lifecycle** section, configure the lifecycle of the container.

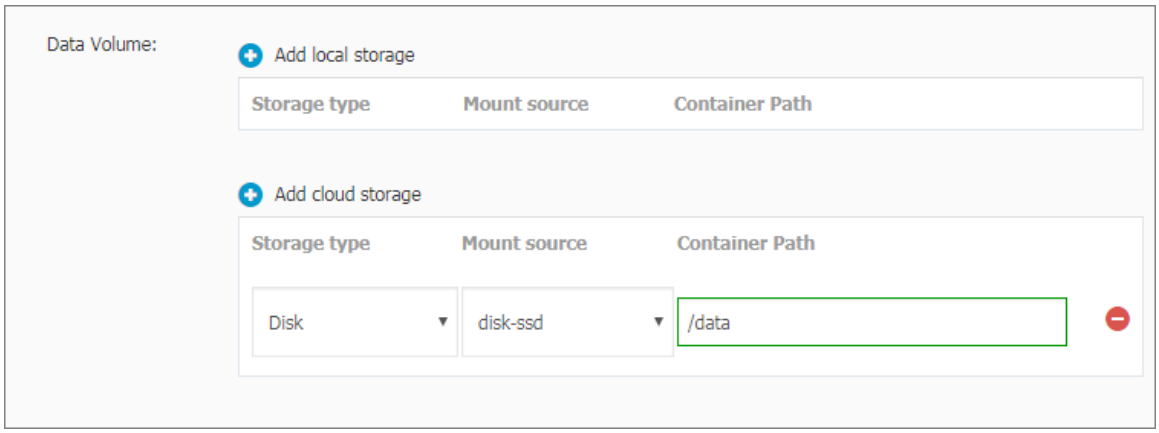
   You can set the following parameters to configure the lifecycle of the container: Start, Post Start, and Pre Stop. For more information, see Configure the lifecycle of a container.

Parameter	Description
Start	Specify a command and parameter that take effect before the container starts.
Post Start	Specify a command that takes effect after the container starts.
Pre Stop	Specify a command that takes effect before the container stops.

6. (Optional)In the **Volume** section, you can mount local volumes and persistent volume claims (PVCs) to the container.

Parameter	Description
Add Local Storage	You can select HostPath, ConfigMap, Secret, or EmptyDir from the PV Type drop-down list. Then, set Mount Source and Container Path to mount the volume to the container. For more information, see Volumes.
Add PVC	You can mount persistent volumes (PVs) by using persistent volume claims (PVCs). You must create a PVC before you can select the PVC to mount a PV. For more information, see Create a PVC.

In this example, a PVC named disk-ssd is mounted to the */tmp* path of the container.



7. (Optional)In the **Log** section, you can specify logging configurations and add custom tags to the collected log.

> **Notice** Make sure that the Log Service agent is installed in the cluster.

Parameter	Description
Collection Configuration	Logstore: Create a Logstore in Log Service to store log data.
	Log Path in Container: Specify stdout or a container path to collect log data. ○ Collect stdout files: If you specify stdout, the stdout files are collected. ○ Collect log from a path: If you specify a path of the container, log is collected from the specified path. In this example, */var/log/nginx* is specified as the path. Wildcard characters can be used in the path.
Custom Tag	You can also add custom tags. The tags are added to the log of the container when the log is collected. Custom tags make it easy to perform statistical analytics and filtering on log data.

8. Click **Next**.

   Proceed to the **Advanced** wizard page.

Step 3: Configure advanced settings

On the **Advanced** wizard page, configure the following settings: access control, scaling, scheduling, annotations, and labels.

1. In the **Access Control** section, you can configure access control settings for exposing backend pods.

> **Note**
> You can configure the following access control settings based on your business requirements:
>
> - Internal applications: For applications that provide services within the cluster, you can create a **ClusterIP** or **NodePort** Service to enable internal communication.
> - External applications: For applications that are exposed to the Internet, you can configure access control by using one of the following methods:
>   - Create a LoadBalancer Service. When you create a Service, set Type to **Server Load Balancer**. You can select or create a Server Load Balancer (SLB) instance for the Service and use the Service to expose your application to the Internet.
>   - Create an Ingress and use it to expose your application to the Internet. For more information, see Ingress.

You can specify how backend pods are exposed. In this example, a ClusterIP Service and an Ingress are created to expose the NGINX application to the Internet.

- To create a Service, click **Create** on the right side of **Services**. In the Create Service dialog box, set the parameters.

Parameter	Description
Name	The name of the Service. In this example, nginx-svc is used.
	The type of Service. This parameter determines how the Service is accessed. ■ **Cluster IP**: The ClusterIP type Service. This type of Service exposes the Service by using an internal IP address of the cluster. If you select this type, the Service is accessible only within the cluster. This is the default type. **Note** The **Headless Service** check box is available only when you set Type to **Cluster IP**. ■ **Node Port**: The NodePort type Service. This type of Service is accessed by using the IP address and a static port of each node. A NodePort Service can be used to route requests to a ClusterIP Service. The ClusterIP Service is automatically created by the system. You can access a NodePort Service from outside the cluster by sending requests to `<NodeIP>:<NodePort>`.

Parameter	Description
Type	■ **Server Load Balancer**: The LoadBalancer type Service. This type of Service exposes the Service by using an SLB instance. If you select this type, you can enable internal or external access to the Service. SLB instances can be used to route requests to NodePort and ClusterIP Services. ■ Create SLB Instance: You can click **Modify** to change the specification of the SLB instance. ■ Use Existing SLB Instance: You can select an existing SLB instance. **Note** You can create an SLB instance or use an existing SLB instance. You can also associate an SLB instance with more than one Service. However, you must take note of the following limits: ■ If you use an existing SLB instance, the listeners of the SLB instance overwrite the listeners of the Service. ■ If an SLB instance is created for a Service, you cannot reuse the SLB instance to expose other Services. Otherwise, the SLB instance may be accidentally deleted. Only SLB instances that are manually created in the console or by calling the API can be reused. ■ An SLB instance must listen on different Service ports if the SLB instance exposes more than one Service. Otherwise, port conflicts may occur. ■ When you reuse an SLB instance, the names of listeners and vServer groups are used as unique identifiers in Kubernetes. Do not modify the names of listeners and vServer groups. ■ You cannot use an SLB instance to expose Services across clusters. **Cluster IP** is selected in this example.
Port Mapping	Specify a Service port and a container port. The container port must be the same as the one that is exposed in the backend pod.
External Traffic Policy	■ Local: Traffic is routed to only the node where the Service is deployed. ■ Cluster: Traffic can be routed to pods on other nodes. **Note** The **External Traffic Policy** parameter is available only when you set Type to **Node Port** or **Server Load Balancer**.
Annotations	Add an annotation to the Service to modify the configurations of the SLB instance. For example, `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20` specifies that the maximum bandwidth of the Service is 20 Mbit/s. This limits the amount of traffic that flows through the Service. For more information, see Use annotations to configure load balancing.
Label	Add a label to the Service.

- To create an Ingress, click **Create** on the right side of **Ingresses**. In the Create dialog box, set the parameters.
  For more information about how to configure an Ingress, see Basic operations of an Ingress.

Notice When you deploy an application from an image, you can create an Ingress for only one Service. In this example, a virtual hostname is specified as the test domain name. You must add a mapping to the *hosts* file for this domain name in the following format: *External endpoint of the Ingress* + *domain name of the Ingress*. In actual scenarios, use a domain name that has obtained an Internet Content Provider (ICP) number.

```
101.37.XX.XX  foo.bar.com  # The IP address of the Ingress.
```

Parameter	Description
Name	The name of the Ingress. In this example, nginx-ingress is used.
Rules	Ingress rules are used to enable access to specified Services in a cluster. For more information, see Configure an Ingress. ■ **Domain**: Enter the domain name of the Ingress. ■ **Path**: Enter the Service URL. The default path is the root path /. The default path is used in this example. Each path is associated with a backend Service. SLB forwards traffic to a backend Service only when inbound requests match the domain name and path. ■ **Services**: Select a Service and a Service port. ■ **EnableTLS**: Select this check box to enable TLS. For more information, see Ingress高级用法. The test domain name **foo.bar.com** is used in this example. The nginx-svc Service is set as the backend of the Ingress.
Weight	Set the weight for each Service in the path. Each weight is calculated as a percentage value. Default value: 100.

Parameter	Description
Canary Release	After a canary release rule is configured, only requests that match the rule are routed to the Service of the new application version. If the weight percentage of a Service is less than 100%, requests that match the rule are routed to the Service based on the weight. ACK supports multiple traffic splitting methods that are applicable to various scenarios, such as canary releases and A/B testing. These methods include: ■ Traffic splitting based on request headers ■ Traffic splitting based on cookies ■ Traffic splitting based on query parameters **Note** Only Ingress controllers of version 0.12.0-5 and later support traffic splitting. The following parameters are required: ■ **Services**: the Service to be accessed. ■ **Type**: the type of matching rule, such as Header, Cookie, and Query. ■ **Name** and **Match Value**: user-defined request parameters that are specified in key-value pairs. ■ **Matching Rule**: Regular expressions and exact matches are supported.
Annotations	■ Click **Rewrite Annotation** to add a rewrite annotation for the Ingress. For example, `nginx.ingress.kubernetes.io/rewrite-target:/` specifies that */path* is redirected to the root path */*. The root path can be recognized by the backend Service. ■ You can also click **Add**, and enter the name and value of an annotation. For more information, see Annotations.
Labels	Add labels to describe the characteristics of the Ingress.

You can find the created Service and Ingress in the **Access Control** section. You can click **Update** or **Delete** to change the configurations.

2. (Optional)In the **Scaling** section, specify whether to enable **HPA** and **CronHPA**. This allows you to meet the resource requirements of the application at different load levels.
   - Horizontal Auto Scaler (HPA) can automatically scale the number of pods in an ACK cluster based on the CPU and memory usage.

Note To enable HPA, you must configure resources that can be scaled for containers. Otherwise, HPA does not take effect.

Parameter	Description
Metric	Select CPU Usage or Memory Usage. The selected resource type must be the same as that specified in the Required Resources field.

Parameter	Description
Condition	Specify the resource usage threshold. The container is scaled out when the threshold is exceeded.
Max. Replicas	Specify the maximum number of pod replicas to which the application can be scaled.
Min. Replicas	Specify the minimum number of pod replicas that must run.

- Cron Horizontal Auto Scaler (CronHPA) can scale an ACK cluster at a scheduled time. Before you enable CronHPA, you must install ack-kubernetes-cronhpa-controller. For more information about CronHPA, see Create CronHPA jobs.

3. (Optional)In the **Scheduling** section, you can configure the following parameters: **Update Method**, **Node Affinity**, **Pod Affinity**, and **Pod Anti Affinity**. For more information, see Affinity and anti-affinity.

> **Note** During pod scheduling, the node labels and pod labels determine the affinities of the node and pod. You can configure node affinities and pod affinities by using preset labels or by using labels that are manually added.

Parameter	Description
Update Method	After you select Enable, you can select Rolling Update or OnDelete. For more information, see Deployments.
Node Affinity	Add labels to worker nodes to set **Node Affinity**. Node affinity supports required and preferred rules, and various operators, such as In, NotIn, Exists, DoesNotExist, Gt, and Lt. - **Required**: Specify the rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution field of the nodeAffinity parameter. These rules have the same effect as the `NodeSelector` parameter. In this example, pods can be scheduled to only nodes with the specified labels. You can create multiple required rules. However, only one of them must be met. - **Preferred**: Specify the rules that are not required to be matched for pod scheduling. Pods are scheduled to a node that matches the preferred rules when multiple nodes match the required rules. In the YAML file, these rules are defined by the preferredDuringSchedulingIgnoredDuringExecution field of the nodeAffinity parameter. In this example, the scheduler attempts to schedule a pod to a node that matches the preferred rules. You can set node weights in preferred rules. If multiple nodes match the required and preferred rules, the node with the highest weight is preferred for pod scheduling. You can create multiple preferred rules. However, all of them must be met before the pod can be scheduled.

Parameter	Description
Pod Affinity	Pod affinity specifies that pods can be scheduled to nodes or topological domains where pods with matching labels are deployed. For example, you can use pod affinity to deploy services that communicate with each other to the same topological domain, such as a host. This reduces the network latency between these services. Pod affinity enables you to select nodes to which pods can be scheduled based on the labels of other running pods. Pod affinity supports required and preferred rules, and the following operators: **In, NotIn, Exists, and DoesNotExist**. ○ **Required**: Specify rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution field of the podAffinity parameter. A node must match the required rules before pods can be scheduled to the node. ■ **Namespace**: Specify the namespace to apply the required rule. Pod affinity rules are defined based on the labels that are added to pods and therefore must be scoped to a namespace. ■ **Topological Domain**: Set the topologyKey. This specifies the key for the node label that the system uses to denote the topological domain. For example, if you set the parameter to **kubernetes.io/hostname**, topologies are determined by nodes. If you set the parameter to **beta.kubernetes.io/os**, topologies are determined by the operating systems of nodes. ■ **Selector**: Click Add to add required labels. ■ **View Applications**: Click **View Applications** and set the namespace and application in the dialog box that appears. You can view the pod labels on the selected application and select the labels as selectors. ■ Required Rules: Specify labels of existing applications, the operator, and the label value. In this example, the required rule specifies that the application to be created is scheduled to a host that runs applications with the **app:nginx** label. ○ **Preferred**: Specify rules that are not required to be matched for pod scheduling. In the YAML file, preferred rules are defined by the preferredDuringSchedulingIgnoredDuringExecution field of the podAffinity parameter. The scheduler attempts to schedule the pod to a node that matches the preferred rules. You can set node weights in preferred rules. Configure the other parameters as described in the preceding settings. **Note** **Weight**: Set the weight in a preferred rule to a value from 1 to 100. The scheduler calculates the weight of each node that meets the preferred rule based on an algorithm, and then schedules the pod to the node with the highest weight.

Parameter	Description
Pod Anti Affinity	Pod anti-affinity specifies that pods are not scheduled to topological domains where pods with matching labels are deployed. Pod anti-affinity rules apply to the following scenarios: ○ Schedule the pods of an application to different topological domains, such as multiple hosts. This allows you to enhance the stability of the service. ○ Grant a pod exclusive access to a node. This enables resource isolation and ensures that no other pod can share the resources of the specified node. ○ Schedule pods of an application to different hosts if the pods may interfere with each other. **Note** The parameters of pod anti-affinity rules are the same as those of pod affinity rules. You can create the rules for different scenarios.
Toleration	Configure tolerations to allow pods to be scheduled to nodes with matching taints.
Schedule to Virtual Nodes	Specify whether to schedule pods to virtual nodes. This parameter is unavailable if the cluster does not contain a virtual node.

4. (Optional)In the **Labels and Annotations** section, click **Add** to configure labels and annotations for the pod.

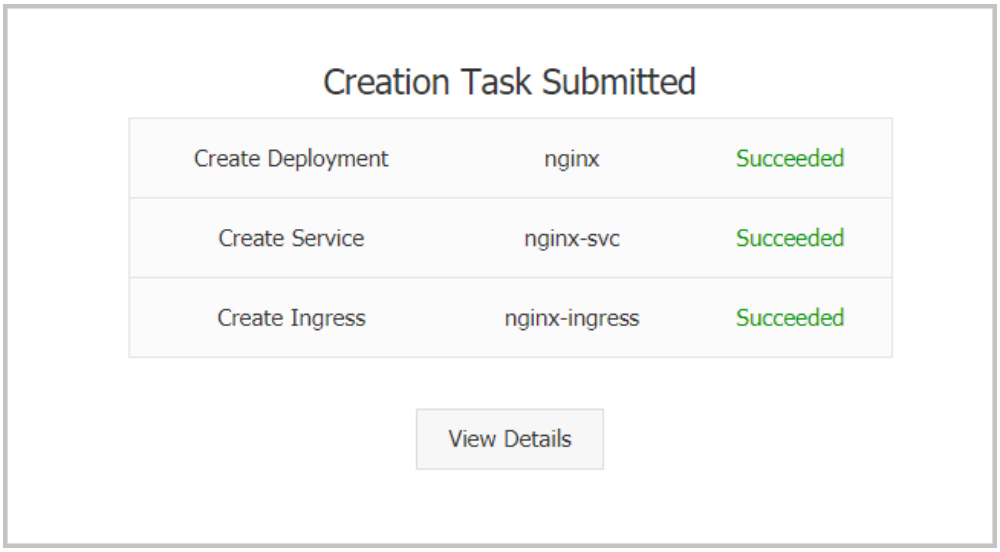
Parameter	Description
Pod Labels	Add a label to the pod. The label is used to identify the application.
Pod Annotations	Add an annotation to the pod configurations.

5. Click **Create**.

Step 4: Check the application

On the **Complete** wizard page, you can view the created application.

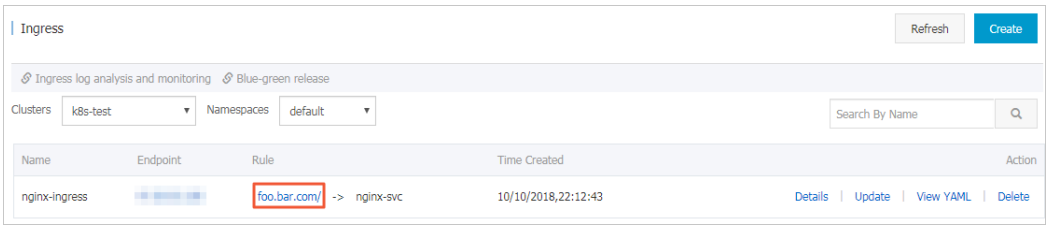
1. Click **View Details** below **Creation Task Submitted**.



The nginx-deployment details page appears.

> **Note** You can also perform the following steps to create an Ingress and a Service: Go to the **Access Method** tab of the application details page, as shown in the preceding figure.
>
> - Click **Create** on the right side of Services to create a Service.
> - Click **Create** on the right side of Ingresses to create an Ingress.

2. In the left-side navigation pane, choose **Services and Ingresses > Ingresses**. On the Ingresses page, you can find the created Ingress.



3. Enter the test domain name in the address bar of your browser and press Enter. The NGINX welcome page appears.

What to do next

View application details

In the left-side navigation pane of the ACK console, click **Clusters**. Click the name of the cluster where the application is deployed or click **Details** in the **Actions** column. Choose **Workloads > Deployments**. On the **Deployments** page, click the name of the deployed application or click **Details** in the **Actions** column.

> **Note** On the Deployments page, you can click **Label**, enter the key and `value` of a label added to the application, and then click **OK** to filter the Deployments.

On the details page of the application, you can view the YAML file of the application. You can also edit, scale, redeploy, and refresh the application.

Action	Description
Edit	On the details page of the application, click **Edit** in the upper-right corner of the page to modify the application configurations.
Scale	On the details page of the application, click **Scale** in the upper-right corner of the page to scale the application to a required number of pods.
View in YAML	On the details page of the application, click **View in YAML** to **update** or **download** the YAML file. You can also click **Save As** to save the file as a different name.

Action	Description
Redeploy	On the details page of the application, click **Redeploy** in the upper-right corner of the page to redeploy the application.
Refresh	On the details page of the application, click **Refresh** in the upper-right corner of the page to refresh the application details page.

Manage the application

On the **Deployments** page, select the application and click **More** in the **Actions** column to perform the following operations.

Action	Description
View in YAML	View the YAML file of the application.
Redeploy	Redeploy the application.
Edit Label	Configure the labels of the application.
Node Affinity	Configure the node affinity rules of the application. For more information, see Scheduling.
Scaling	Configure the scaling settings of the application. For more information, see HPA and CronHPA.
Toleration	Configure the toleration rules of the application. For more information, see Scheduling.
Upgrade Policy	Configure the upgrade policy of the application. • **Rolling Update**: Pods are updated in a rolling update fashion. • **OnDelete**: All existing pods are deleted before new pods are created.
Clone	Create a new application by using the same container settings as the current application.
Roll Back	Roll back the application to a previous version.
Logs	View the log data of the application.
Delete	Delete the application.

Create a Linux application by using an orchestration template

In an orchestration template, you must define the resources that are required for running an application and use mechanisms such as label selectors to manage these resources.

This section describes how to use an orchestration template to create an NGINX application that consists of a Deployment and a Service. The Deployment provisions pods for the application and the Service manages access to the pods at the backend.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner of the page.
6. Configure the parameters and click **Create**.
   - **Namespace**: Select the namespace to which the resource objects belong. By default, the default namespace is selected. Most resources are scoped to namespaces, except for underlying computing resources, such as nodes and persistent volumes (PVs).
   - **Sample Template**: ACK provides YAML templates of various resource types. This simplifies the deployment of resource objects. You can also create a custom template based on YAML syntax to define the resources that you want to create.
   - **Add Deployment**: This feature allows you to define a YAML template.
   - **Use Existing Template**: You can import an existing template to the configuration page.
   - **Save As**: You can save the template that you have configured.

   The following sample template is based on an orchestration template provided by ACK. You can use this template to create a Deployment to run an NGINX application.

   > **Note**
   > - ACK supports YAML syntax. You can use the `---` symbol to separate multiple resource objects. This allows you to create multiple resource objects in a single template.
   > - (Optional)By default, if you mount a volume to the application, the existing files in the mount path of the application are overwritten. To avoid the existing files from being overwritten, you can add a `subPath` parameter.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment-basic
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
        ports:
        - containerPort: 80
        volumeMounts:
          - name: nginx-config
            mountPath: /etc/nginx/nginx.conf
            subPath: nginx.conf  # Set the subPath parameter.
    volumes:
      - name: nginx-config
        configMap:
          name: nginx-conf
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: my-service1      #TODO: to specify your service name
  labels:
    app: nginx
spec:
  selector:
    app: nginx          #TODO: change label selector to match your backend pod
  ports:
  - protocol: TCP
    name: http
    port: 30080         #TODO: choose an unique port on each node to avoid port conflict
    targetPort: 80
  type: type: LoadBalancer      ## In this example, the Service type is changed from Nodeport to LoadBalancer.

---
# The ConfigMap of the mounted volume.
apiVersion: v1
kind: ConfigMap
metadata:
 name: nginx-conf
 namespace: default
data:
 nginx.conf: |-
  user  nginx;
  worker_processes  1;
  error_log  /var/log/nginx/error.log warn;
  pid      /var/run/nginx.pid;
  events {
     worker_connections  1024;
  }
  http {
     include     /etc/nginx/mime.types;
     default_type  application/octet-stream;
     log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                 '$status $body_bytes_sent "$http_referer" '
                 '"$http_user_agent" "$http_x_forwarded_for"';
     access_log  /var/log/nginx/access.log  main;
     sendfile      on;
     #tcp_nopush    on;
     keepalive_timeout  65;
     #gzip  on;
     include /etc/nginx/conf.d/*.conf;
  }
```

7. Click **Create**. A message that indicates the deployment status appears.

Manage applications by using commands

This topic describes how to create an application or view containers of an application by using commands.

Create an application by using commands

1. In a CLI, run the following command to start a container. An NGINX web server is used in this example.

```
kubectl run -it nginx --image=registry.aliyuncs.com/spacexnice/netdia:latest
```

2. Run the following command to create a service entry for this container. The `--type=LoadBalancer` parameter in the command indicates that the system must create a Server Load Balancer (SLB) instance for the NGINX container.

```
kubectl expose deployment nginx --port=80 --target-port=80 --type=LoadBalancer
```

View containers by using commands

In the CLI, run the `kubectl get pods` command to list all running containers in the default namespace.

```
NAME                READY   STATUS   RESTARTS  AGE
nginx-2721357637-d****      1/1     Running  1     9h
```

Create an application by using an image pull Secret

Container Service for Kubernetes (ACK) allows you to use image pull Secrets in the ACK console. You can create an image pull Secret or use an existing image pull Secret.

When you create an application from a private image, you must set a Secret for image pulling to ensure the security of the image. In the ACK console, you can specify the authentication information of the private image repository as a Secret of the docker-registry type. This Secret applies to the specified Kubernetes cluster.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, select the namespace and click **Create from Image** in the upper-right corner of the page.
6. On the **Basic Information** wizard page, set the parameters. For more information, see Basic Information.
7. Configure containers.

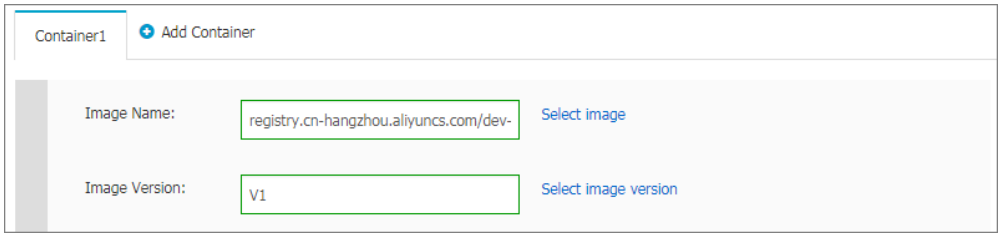
   The following example describes how to configure an image pull Secret. For more information about container configurations, see Configure containers.

   i. On the **Container** wizard page, enter the address of the private image in the **Image Name** field. The address must be in the following format: `domainname/namespace/imagename`.

   > **Note** Public images do not require Secrets.

   ii. Enter the image version that you want to use in the Image Version field.

   A Tomcat private image is used in this example.

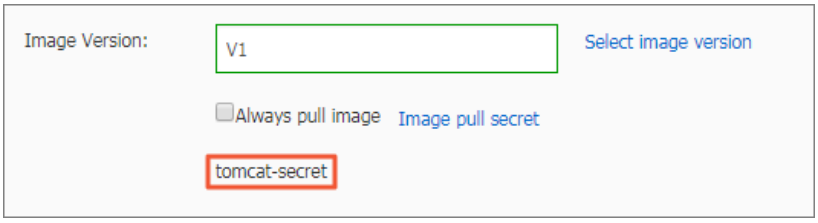


iii. Click **Set Image Pull Secret**, and create a Secret or select an existing Secret in the dialog box that appears.

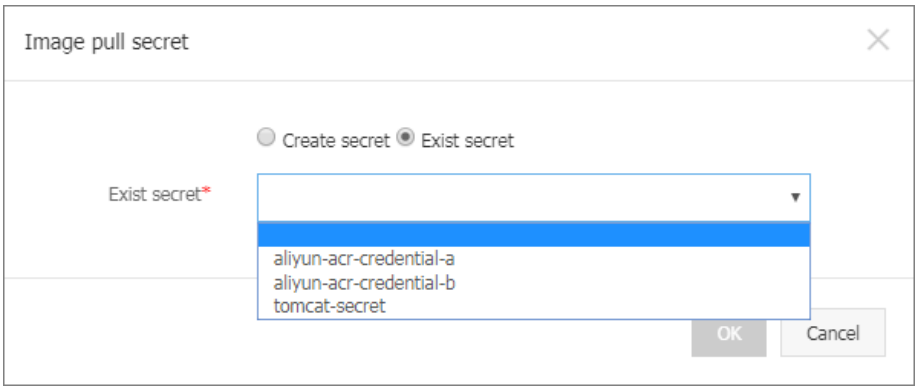
- You can select **New Secret**. If you select New Secret, set the following parameters:

Parameter	Description
Name	The key name of the Secret. You can enter a custom name.
Repository Domain	The address of the specified Docker registry. If you use an image repository in Container Registry, the address of the image repository is automatically displayed.
Username	The username used to log on to the Docker repository. If you use an image repository in Container Registry, your account name is used as the username. Alibaba Cloud accounts and Resource Access Management (RAM) users are supported.
Password	The password used to log on to the Docker repository. If you use an image repository in Container Registry, the password is the logon password of Container Registry.
Email	The email address. This parameter is optional.

Click **OK**. The newly created Secret appears on the page.



- You can also select **Existing Secret**. You can use commands or YAML files to create image pull Secrets. For more information, see How do I use private images in Kubernetes clusters? and Create an application from a private image repository.



8. Configure other parameters based on your requirements. Then, click **Create**.

   For more information, see Step 3: Configure advanced settings.

9. In the left-side navigation pane of the ACK console, click **Clusters**.
10. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
11. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

12. On the **Deployments** page, check the status of the application.
    Verify that the Tomcat application runs as expected. This indicates that the Tomcat private image is pulled by using the image pull Secret.

References

- Use a StatefulSet to create a stateful application
- Use annotations to configure load balancing

7.1.2. Use a StatefulSet to create a stateful application

Container Service for Kubernetes (ACK) allows you to create stateful applications by using the ACK console. This topic provides an example on how to create a stateful NGINX application and demonstrates the features of StatefulSets.

Prerequisites

Before you deploy a stateful application from an image, make sure that you have performed the following steps:

- 创建Kubernetes托管版集群
- Create a PVC
- Connect to Kubernetes clusters by using kubectl

Background information

StatefulSets provide the following features:

Feature	Description
Pod consistency	Pod consistency ensures that pods are started and terminated in the specified order and ensures network consistency. Pod consistency is determined by pod configurations, regardless of the node to which a pod is scheduled.
Stable and persistent storage	VolumeClaimTemplate allows you to mount a persistent volume (PV) to each pod. The mounted PVs are not deleted after you delete or scale in the number of pod replicas.
Stable network identifiers	Each pod in a StatefulSet derives its hostname from the name of the StatefulSet and the ordinal of the pod. The pattern of the hostname is `StatefulSet name-pod ordinal`.
Stable orders	For a StatefulSet with N pod replicas, each pod is assigned an integer ordinal from 0 to N-1. The ordinals assigned to pods within the StatefulSet are unique.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > StatefulSets**.

5. In the upper-right corner of the **StatefulSets** page, click **Create from Image**.
6. On the **Basic Information** wizard page, configure the basic settings.

   In this example, the Type parameter is set to **StatefulSet** to deploy a stateful application.

Parameter	Description
Name	The name of the application.
Namespace	The namespace where you want to deploy the application. The default namespace is automatically selected. You can select another namespace.
Replicas	The number of pods that are provisioned for the application.
Type	The type of the application. You can select **Deployment**, **StatefulSet**, **Job**, **CronJob**, or **DaemonSet**.
Label	Add a label to the application. The label is used to identify the application.
Annotations	Add an annotation to the application.
Synchronize Timezone	Specify whether to synchronize the time zone between nodes and containers.

7. Click **Next** to proceed to the **Container** wizard page.
8. Configure containers.

   > **Note** In the upper part of the **Container** wizard page, click **Add Container** to add more containers for the application.

   The following table describes the parameters that are required to configure the containers.

   - General settings

Parameter	Description

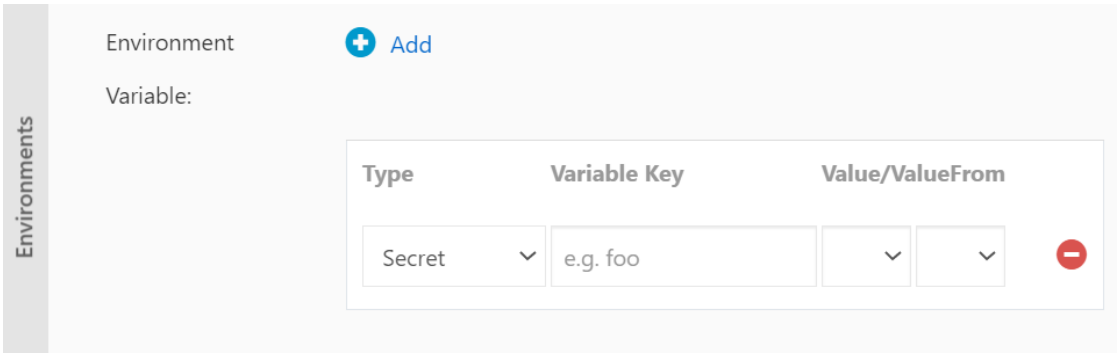
Parameter	Description
Image Name	■ You can click **Select Image**. In the dialog box that appears, select an image and click **OK**. In this example, an NGINX image is selected. On the **Search** tab, select **Docker Images** from the drop-down list, enter *NGINX* into the search box, and then click Search. ■ Images from Container Registry: On the **Alibaba Cloud Container Registry** tab, you can select an image from Container Registry. You must select the region and the Container Registry instance to which the image belongs. For more information about Container Registry, see What is Container Registry?. **Note** On the **Alibaba Cloud Container Registry** tab, you can search for images by name. ■ Docker Official Images: On the **Docker Official Images** tab, you can select a Docker image. ■ Favorite Images: On the **Favorite Images** tab, you can select a Docker image that you have added to your favorite list. ■ Search: On the **Search** tab, you can select Alibaba Cloud Image from the drop-down list and specify a region to search for an image in Container Registry. You can also select Docker Images from the drop-down list and search for a Docker image. ■ You can also enter the address of a private registry. The registry address must be in the `domainname/namespace/imagename:tag` format.

Parameter	Description
Image Version	■ Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used. ■ You can select the following image pull policies: ■ **ifNotPresent**: If the image that you want to pull is found on your on-premises machine, the image on your on-premises machine is used. Otherwise, ACK pulls the image from the corresponding repository. ■ **Always**: ACK pulls the image from Container Registry each time the application is deployed or scaled out. ■ **Never**: ACK uses only images on your on-premises machine. **Note** If you select **Image Pull Policy**, no image pull policy is applied. ■ To pull the image without a password, click **Set Image Pull Secret** to set a Secret that is used to pull the image. For more information, see Use aliyun-acr-credential-helper to pull images without a password.
Resource Limit	You can specify an upper limit for the CPU, memory, and ephemeral storage space that the container can consume. This prevents the container from occupying an excess amount of resources. The CPU resource is measured in millicores (one thousandth of one core). The memory resource is measured in MiB. The ephemeral storage resource is measured in GiB.
Required Resources	The amount of CPU and memory resources that are reserved for this application. These resources are exclusive to the container. This prevents the application from becoming unavailable if other services or processes compete for computing resources.
Container Start Parameter	■ stdin: Pass stdin to the container. ■ tty: Stdin is a TeleTYpewriter (TTY).
Privileged Container	■ If you select Privileged Container, privileged=true is set for the container and the privilege mode is enabled. ■ If you do not select Privileged Container, privileged=false is set for the container and the privilege mode is disabled.
Init Container	If you select Init Container, an init container is created. An init container contains useful utilities. For more information, see Init Containers.

- (Optional)Ports
  Configure container ports.

- Name: Enter a name for the port.
- Container Port: Enter the container port that you want to open. Enter a port number from 1 to 65535.
- Protocol: Select TCP or UDP.

- (Optional)Environments
  You can configure environment variables for pods in key-value pairs. Environment variables are used to apply pod configurations to containers. For more information, see Pod variables.
  - Type: Select the type of environment variable. You can select **Custom**, **ConfigMaps**, **Secrets**, or **Value/ValueFrom**. If you select ConfigMaps or Secrets as the type of environment variable, all values in the selected ConfigMap or Secret are passed to the container environment variables. In this example, Secrets is selected.
    Select **Secrets** from the Type drop-down list and select a Secret from the **Value/ValueFrom** drop-down list. All values in the selected Secret are passed to the environment variable.

    In this case, the *YAML* file used to deploy the application contains the settings that reference all values in the specified Secret.

    ```
    envFrom:
      - secretRef:
          name: test
    ```

  - Variable Key: Specify the key of the environment variable.
  - Value/ValueFrom: Specify the value that is referenced by the environment variable.

- (Optional)Health Check
  Health check settings include liveness and readiness probes. Liveness probes determine when to restart the container. Readiness probes determine whether the container is ready to accept network traffic. For more information about health checks, see Configure Liveness, Readiness, and Startup Probes.

Request type	Description

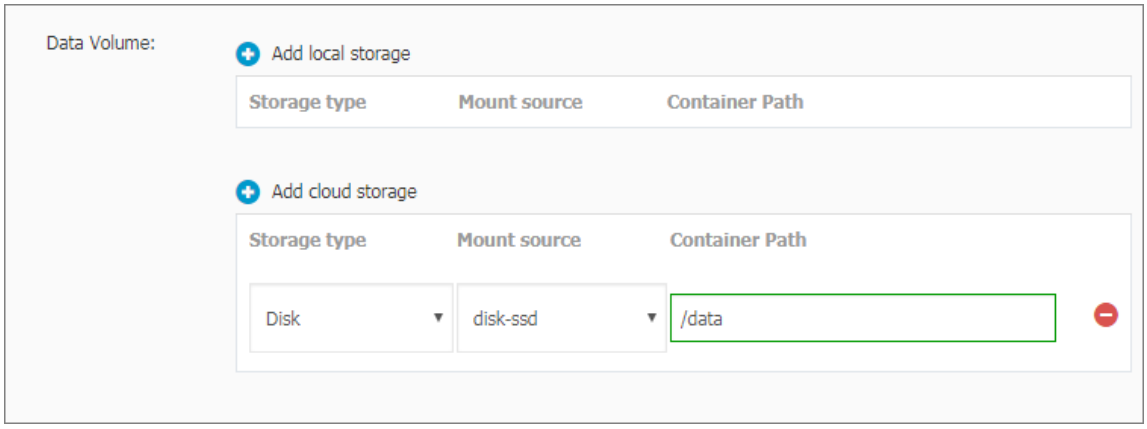
Request type	Description
HTTP	Sends an HTTP GET request to the container. You can configure the following parameters: ■ Protocol: HTTP or HTTPS. ■ Path: Enter the requested path on the server. ■ Port: Enter the container port that you want to open. Enter a port number from 1 to 65535. ■ HTTP Header: Enter the custom headers in the HTTP request. Duplicate headers are allowed. Key-value pairs are supported. ■ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the wait time (in seconds) before the first probe is performed after the container is started. Default value: 3. ■ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ■ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which the probe times out. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of consecutive successes that must occur before a container is considered healthy after a failed probe. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of consecutive failures that must occur before a container is considered unhealthy after a success. Default value: 3. Minimum value: 1.
TCP	Sends a TCP socket to the container. kubelet attempts to open the socket on the specified port. If the connection can be established, the container is considered healthy. Otherwise, the container is considered unhealthy. You can configure the following parameters: ■ Port: Enter the container port that you want to open. Enter a port number that ranges from 1 to 65535. ■ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the wait time (in seconds) before the first probe is performed after the container is started. Default value: 15. ■ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ■ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which the probe times out. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of consecutive successes that must occur before a container is considered healthy after a failed probe. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of consecutive failures that must occur before a container is considered unhealthy after a success. Default value: 3. Minimum value: 1.

Request type	Description
Command	Runs a probe command in the container to check the health status of the container. You can configure the following parameters: ■ Command: Enter the probe command that is run to check the health status of the container. ■ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the wait time (in seconds) before the first probe is performed after the container is started. Default value: 5. ■ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ■ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which the probe times out. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of consecutive successes that must occur before a container is considered healthy after a failed probe. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of consecutive failures that must occur before a container is considered unhealthy after a success. Default value: 3. Minimum value: 1.

- Lifecycle
  You can set the following parameters to configure the lifecycle of the container: Start, Post Start, and Pre Stop. For more information, see Attach Handlers to Container Lifecycle Events.
  - **Start**: Set the command and parameter that take effect before the container starts.
  - **Post Start**: Set the command that takes effect after the container starts.
  - **Pre Stop**: Set the command that takes effect before the container stops.

- (Optional)Volume
  You can mount local volumes and persistent volume claims (PVCs) to the container.
  - **Add Local Storage**: You can select HostPath, ConfigMap, Secret, and EmptyDir. The specified volume is mounted to a path in the container. For more information, see Volumes.
  - **Add PVC**: You can select Cloud Storage.

In this example, a PVC named disk-ssd is mounted to the */tmp* path of the container.



- (Optional)Log
  Configure **Log Service**. You can specify collection configurations and add custom tags.

  > **Notice** Make sure that the Log Service agent is installed in the cluster.

Parameter	Description
Collection Configuration	Logstore: creates a Logstore in Log Service to store collected logs.
	Log Path in Container: specifies stdout or a path to collect logs. ■ stdout: specifies that the stdout files are collected. ■ Text Logs: specifies that logs in the specified path of the container are collected. In this example, */var/log/nginx* is specified as the path. Wildcard characters can be used in the path.
Custom Tag	You can also add custom tags. Custom tags are added to the log of the container when the log is collected. Custom tags provide an easy method to filter collected logs and perform statistical analytics.

9. Set the parameters based on your business requirements and click **Next**.
10. (Optional)Configure advanced settings.
    - Access Control

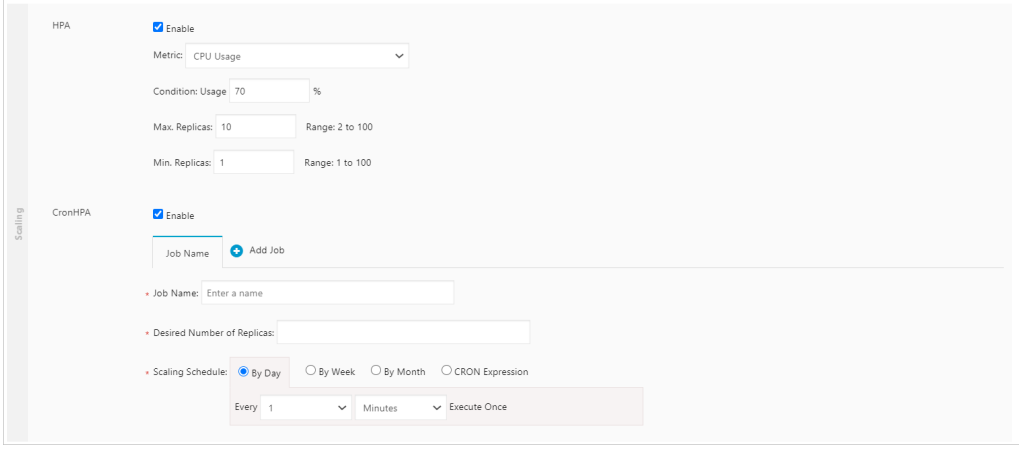
      > **Note**
      > You can configure the following access control settings based on your business requirements:
      > - Internal applications: For applications that run inside the cluster, you can create a ClusterIP or NodePort Service to enable internal communication.
      > - External applications: For applications that are exposed to the Internet, you can configure access control by using one of the following methods:
      >   - Create a LoadBalancer Service and enable access to your application over the Internet by using a Server Load Balancer (SLB) instance.
      >   - Create an Ingress and use the Ingress to expose your application to the Internet. For more information, see Ingress.

You can also specify how the backend pods are exposed to the Internet. In this example, a ClusterIP Service and an Ingress are created to expose the NGINX application to the Internet.

Parameter	Description
Services	Click **Create** on the right side of **Service**. In the **Create Service** dialog box, set the parameters. For more information about the parameters, see Manage Services. **Cluster IP** is selected in this example.
Ingresses	Click **Create** on the right side of **Ingresses**. In the Create dialog box, set the parameters. For more information about how to configure an Ingress, see Create an Ingress. **Note** When you deploy an application from an image, you can create an Ingress only for one Service. In this example, a virtual hostname is used as the test domain name. You must add the following entry to the hosts file to map the domain name to the IP address of the Ingress. In actual scenarios, use a domain name that has obtained an ICP number. `101.37.224.146 foo.bar.com #The IP address of the Ingress.`

You can find the created Service and Ingress in the **Access Control** section. You can click **Update** or **Delete** to change the configurations.

- Scaling
  In the **Scaling** section, specify whether to enable **HPA** and **CronHPA**. Horizontal Pad Autoscaler (HPA) allows you to meet the resource requirements of the application at different load levels.



- HPA can automatically scale the number of pods in an ACK cluster based on the CPU and memory usage.

  > **Note** To enable HPA, you must configure required resources for the container. Otherwise, HPA does not take effect.

Parameter	Description
Metric	Select CPU Usage or Memory Usage. The selected resource type must be the same as that specified in the Required Resources field.
Condition	Specify the resource usage threshold. HPA triggers scale-out activities when the threshold is exceeded.
Max. Replica	Specify the maximum number of pod replicas to which the application can be scaled.
Min. Replica	Specify the minimum number of pod replicas that must run.

- CronHPA can scale an ACK cluster at a scheduled time. For more information about CronHPA, see Create CronHPA jobs.

- Scheduling
  You can set the following parameters: Update Method, Node Affinity, Pod Affinity, Pod Anti Affinity, and Toleration. For more information, see Affinity and anti-affinity.

  > **Note** Node affinity and pod affinity affect pod scheduling based on node labels and pod labels. You can add node labels and pod labels that are provided by Kubernetes to configure node affinity and pod affinity. You can also add custom labels to nodes and pods, and then configure node affinity and pod affinity based on these custom labels.

Parameter	Description
Update Method	Select Rolling Update or OnDelete. For more information, see Deployments.

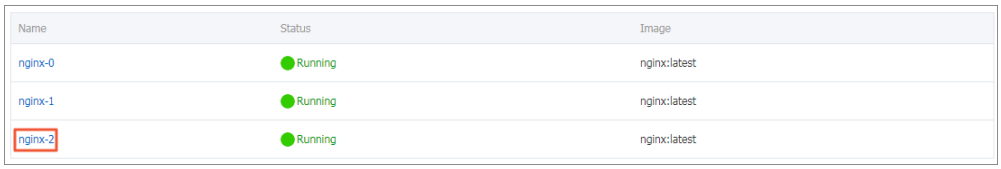
Parameter	Description
Node Affinity	Add labels to worker nodes to set **Node Affinity**. Node Affinity supports required and preferred rules, and various operators, such as In, NotIn, Exists, DoesNotExist, Gt, and Lt. ■ **Required**: Specify the rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution field of the nodeAffinity parameter. These rules have the same effect as the **NodeSelector** parameter. In this example, pods can be scheduled only to nodes with the specified labels. You can create multiple required rules. However, only one of them must be met. ■ **Preferred**: Specify the rules that are not required to be matched for pod scheduling. Pods are scheduled to a node that matches the preferred rules when multiple nodes match the required rules. In the YAML file, these rules are defined by the preferredDuringSchedulingIgnoredDuringExecution field of the nodeAffinity parameter. In this example, the scheduler attempts to not schedule a pod to a node that matches the preferred rules. You can also set weights for preferred rules. If multiple nodes match the rule, the node with the highest weight is preferred. You can create multiple preferred rules. However, all of them must be met before the pod can be scheduled.
Pod Affinity	Pod affinity rules specify how pods are deployed relative to other pods in the same topology domain. For example, you can use pod affinity to deploy services that communicate with each other to the same topological domain, such as a host. This reduces the network latency between these services. Pod affinity enables you to specify to which node pods can be scheduled based on the labels on other running pods. Pod affinity supports required and preferred rules, and the following operators: **In, NotIn, Exists, and DoesNotExist**.

Parameter	Description
	■ **Required**: Specify rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution field of the podAffinity parameter. A node must match the required rules before pods can be scheduled to the node. ■ **Namespace**: Specify the namespace to apply the required rule. Pod affinity rules are defined based on the labels that are added to pods and therefore must be scoped to a namespace. ■ **Topological Domain**: Set the topologyKey. This specifies the key for the node label that the system uses to denote the topological domain. For example, if you set the parameter to `kubernetes.io/hostname`, topologies are determined by nodes. If you set the parameter to `beta.kubernetes.io/os`, topologies are determined by the operating systems of nodes. ■ **Selector**: Click Add to add pod labels. ■ **View Applications**: Click **View Applications** and set the namespace and application in the dialog box that appears. You can view the pod labels on the selected application and add the labels as selectors. ■ Required Rules: Specify labels on existing applications, the operator, and the label value. In this example, the required rule specifies that the application to be created is scheduled to a host that runs applications with the `app:nginx` label. ■ **Preferred**: Specify rules that are not required to be matched for pod scheduling. In the YAML file, preferred rules are defined by the preferredDuringSchedulingIgnoredDuringExecution field of the podAffinity parameter. The scheduler attempts to schedule the pod to a node that matches the preferred rules. You can set weights for preferred rules. The other parameters are the same as those of required rules.
Pod Anti Affinity	Pod anti-affinity rules specify that pods are not scheduled to topological domains where pods with matching labels are deployed. Pod anti-affinity rules apply to the following scenarios: ■ Schedule the pods of an application to different topological domains, such as multiple hosts. This allows you to enhance the stability of the service. ■ Grant a pod exclusive access to a node. This enables resource isolation and ensures that no other pods can share the resources of the specified node. ■ Schedule pods of an application to different hosts if the pods may interfere with each other. **Note** **Weight**: Set the weight of a preferred rule to a value from 1 to 100. The scheduler calculates the weight of each node that meets the preferred rule based on an algorithm, and then schedules the pod to the node with the highest weight. **Note** The parameters of pod anti-affinity rules are the same as those of pod affinity rules. You can create the rules for different scenarios.
Toleration	Configure toleration rules to allow pods to be scheduled to nodes with matching taints.
Schedule to Virtual Nodes	Specify whether to schedule pods to virtual nodes. This parameter is unavailable if the cluster does not contain a virtual node.

- Labels and Annotations
  - Pod Labels: Add a label to the pod. The label is used to identify the application.
  - Pod Annotations: Add an annotation to the pod.

11. Click **Create**.
12. After the application is created, you are redirected to the Complete wizard page. You can find the resource objects under the application and click **View Details** to view application details.

    The details page of the created stateful application appears.
13. In the upper-left corner of the page, click the **Back** icon to go to the StatefulSets page. On the StatefulSets page, you can view the created application.
14. (Optional)Click **Scale** in the Actions column to scale the application.
    i. In the Scale dialog box, set Desired Number of Pods to 3 and click OK. After you scale out the application, all pods in the application are listed in ascending order of ordinal indexes. If you scale in the application, pods are deleted in descending order of ordinal indexes. This ensures that all pods follow a specific order.



    ii. In the left-side navigation pane, choose **Volumes > Persistent Volume Claims**. Verify that after you scale out the application, new PVs and PVCs are created for the newly added pods. However, if the application is scaled in, existing PVs and PVCs are not deleted.

Related operations

In the left-side navigation pane, click **Clusters**. On the Clusters page, click the name of the cluster where the application is deployed or click **Applications** in the **Actions** column. In the left-side navigation pane, choose **Workloads** > **StatefulSets**. On the **StatefulSets** page, click the name of the application that you want to manage or click **Details** in the **Actions** column. On the details page of the application, you can **edit**, **scale**, **redeploy**, and **refresh** the application. You can also **view the YAML file** of the application.

- Edit: On the details page of the application, click **Edit** in the upper-right corner of the page to modify the configurations of the application.
- Scale: On the details page of the application, click **Scale** in the upper-right corner of the page to scale the application to a required number of pods.
- View in YAML: On the details page of the application, click **View in YAML** in the upper-right corner of the page. You can **Update** and **Download** the YAML file. You can also click **Save As** to save the *YAML* file as a different name.
- Redeploy: On the details page of the application, click **Redeploy** in the upper-right corner of the page to redeploy the application.
- Refresh: On the details page of the application, click **Refresh** in the upper-right corner of the page to refresh the application details page.

What's next

Log on to a master node and run the following commands to test persistent storage.

1. Run the following commands to create a temporary file in the disk that is mounted to pod nginx-1:

```
kubectl exec nginx-1 ls /tmp        #Query files in the /tmp directory.
kubectl exec nginx-1 touch /tmp/statefulset      #Create a file named statefulset.
kubectl exec nginx-1 ls /tmp
lost+found
statefulset
```

2. Run the command to delete pod nginx-1 and verify data persistence:

```
kubectl delete pod nginx-1
pod"nginx-1" deleted
```

3. After the system recreates and starts a new pod, query the files in the /tmp directory. The following result shows that the statefulset file still exists. This shows the high availability of the stateful application.

```
kubectl exec nginx-1 ls /tmp                #Query files in the /tmp directory.
statefulset
```

7.1.3. Create a DaemonSet

A DaemonSet ensures that each node runs a copy of a pod. You can use a DaemonSet to run a log collection daemon, a monitoring daemon, or a system management application on each node. This topic describes how to create a DaemonSet for a Container Service for Kubernetes (ACK) cluster.

Create a DaemonSet in the ACK console

Create a DaemonSet from an image

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > DaemonSets**.
5. In the upper-right corner of the **DaemonSets** page, click **Create from Image**.
6. Set parameters for the DaemonSet.
   i. On the **Basic Information** wizard page, configure the basic settings. For more information, see Create a Deployment from an image.
   ii. On the **Container** wizard page, configure one or more containers. For more information, see Create a stateless application by using a Deployment.
   iii. On the **Advanced** wizard page, configure the advanced settings.

   A DaemonSet can schedule a pod to a node that is in the Unschedulable state. To run a pod on only a specific node, set node affinity, pod affinity, or toleration rules. For more information, see Create a stateless application by using a Deployment.
7. Click **Create**.
   After the DaemonSet is created, you can view the DaemonSet on the DaemonSets page.

Create a DaemonSet from a YAML template

1. In the upper-right corner of the **DaemonSets** page, click **Create from YAML**.
2. On the **Create** page, configure the DaemonSet in the **Template** section.
3. Click **Create** below the **Template** section.
   After the DaemonSet is created, you can view the DaemonSet on the DaemonSets page.

Create a DaemonSet by using kubectl

Before you use kubectl to create a DaemonSet, you must download kubectl and connect to your cluster by using kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

A DaemonSet can schedule a pod to a node that is in the Unschedulable state. To run a pod on only a specific node, set the following parameters.

Parameter	Description
nodeSelector	A pod is scheduled to only the node with the specified labels.
nodeAffinity	Node affinity. Pods are scheduled to nodes based on node labels. Node affinity allows you to set other matching rules.
podAffinity	Pod affinity. Pods are scheduled to nodes based on pod labels. A pod is scheduled to only the node that runs a pod that matches the affinity rules.

To demonstrate how to create a DaemonSet by using kubectl, a DaemonSet named fluentd-elasticsearch is created in this example.

1. Create a *daemonset.yaml* file and copy the following content into the file:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: fluentd-elasticsearch
 namespace: kube-system
 labels:
  k8s-app: fluentd-logging
spec:
 selector:
  matchLabels:
   name: fluentd-elasticsearch
 template:
  metadata:
   labels:
    name: fluentd-elasticsearch
  spec:
   tolerations:
   # this toleration is to have the daemonset runnable on master nodes
   # remove it if your masters can't run pods
   - key: node-role.kubernetes.io/master
    effect: NoSchedule
   containers:
   - name: fluentd-elasticsearch
    image: quay.io/fluentd_elasticsearch/fluentd:v2.5.2
    resources:
     limits:
      memory: 200Mi
     requests:
      cpu: 100m
      memory: 200Mi
    volumeMounts:
    - name: varlog
     mountPath: /var/log
    - name: varlibdockercontainers
     mountPath: /var/lib/docker/containers
     readOnly: true
   terminationGracePeriodSeconds: 30
   volumes:
   - name: varlog
    hostPath:
     path: /var/log
   - name: varlibdockercontainers
    hostPath:
     path: /var/lib/docker/containers
```

2. Run the following command to create a DaemonSet:

```
kubectl create -f daemonset.yaml
```

If `daemonset.apps/fluentd-elasticsearch created` is returned, the DaemonSet is created.

What to do next

After you create a DaemonSet, you can perform the following operations:

- On the DaemonSets page, find the created DaemonSet and click **Details** in the **Actions** column. On the details page, you can view basic information about the DaemonSet. The information includes pods, access method, events, and logs.

- On the DaemonSets page, find the created DaemonSet. You can choose **More > View in YAML** in the **Actions** column to view the YAML file of the DaemonSet. You can also choose **More > Delete** to delete the DaemonSet.

7.1.4. Use a Job to create an application

You can use a Job to create an application in the Container Service for Kubernetes (ACK) console. This topic describes how to use a Job to create a busybox application and features of the application.

Prerequisites

An ACK cluster is created. For more information, see Create a dedicated Kubernetes cluster.

Context

A Job processes multiple short-lived, one-off tasks at a time to ensure that one or more pods in the tasks terminate with success.

Kubernetes supports the following types of Jobs:

- Non-parallel Jobs: In most cases, a Job of this type starts only one pod unless the pod fails. The Job is considered complete when the pod terminates with success.
- Jobs with a fixed completion count: A non-zero positive value is specified for `.spec.completions`. A Job of this type starts pods one after one. The Job is considered complete when the number of pods that terminate with success is equal to the value of `.spec.completions`.
- Parallel Jobs with a work queue: A non-zero positive value is specified for `.spec.Parallelism`. A Job of this type starts multiple pods at a time. The Iob is considered complete when all pods terminate and at least one pod terminates with success. `.spec.completions` is not required.
- Parallel Jobs with a fixed completion count: A Job of this type has both `.spec.completions` and `.spec.Parallelism` specified. The Job starts multiple pods at a time to process a work queue.

Jobs can manage pods based on the settings of `.spec.completions` and `.spec.Parallelism` as described in the following table.

Note The Job created in this example is a parallel Job with a fixed completion count.

Job type	Scenario	Action	completions	Parallelism
One-off Job	Migrate a database	The Job starts only one pod. The Job is complete when the pod terminates with success.	1	1
Job with a fixed completion count	Start pods one after one to process a work queue	The Job starts pods one after one. The Job is complete when the number of pods that terminate with success reaches the value of .spec.completions.	2+	1

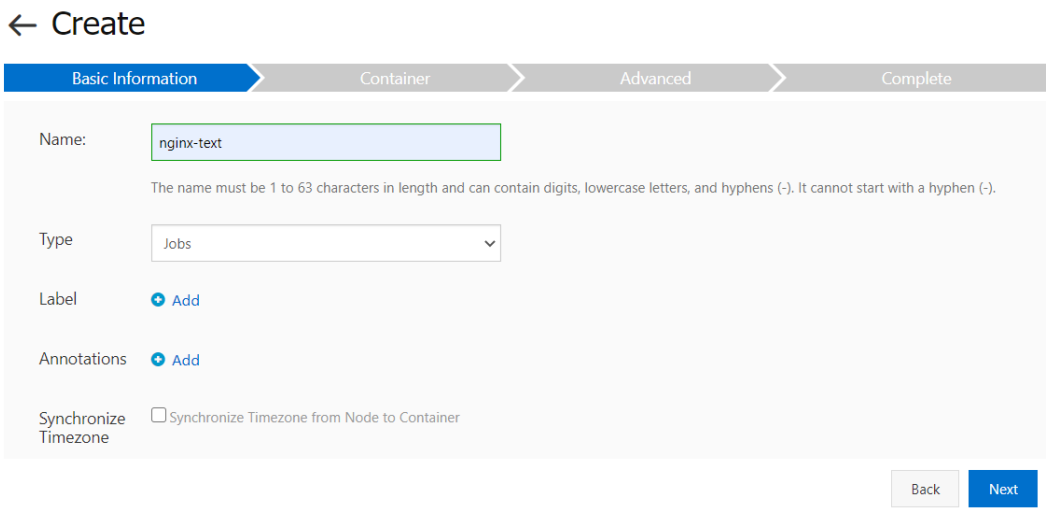
Job type	Scenario	Action	completions	Parallelism
Parallel Job with a fixed completion count	Start multiple pods at a time to process a work queue	The Job starts multiple pods at a time. The Job is complete when the number of pods that terminate with success reaches the value of .spec.completions.	2+	2+
Parallel Job	Start multiple pods at a time to process a work queue	The Job starts one or more pods at a time. The Job is complete when at least one pod terminates with success.	1	2+

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Jobs**.
5. On the **Jobs** page, click **Create from Image** in the upper-right corner of the page.
   - **Name**: the name of the application.
   - **Type**: the application type. Select **Jobs** from the drop-down list.

     > **Note** The application is created by using a **Job** in this example.

   - **Label**: Add a label to the application. The label is used to identify the application.
   - **Annotations**: Add an annotation to the application.
   - **Synchronize Timezone**: Specify whether to synchronize the time zone between nodes and containers.



6. Configure containers.

> **Note** You can configure one or more containers for the pods of the application.

i. Configure basic settings.

- **Image Name**: Select an image.
  - To use a Docker image or an image from Container Registry, click **Select Image**. In the dialog box that appears, select an image and click **OK**. In this example, an NGINX image is selected. On the **Search** tab, you can select **Docker Images** from the drop-down list, enter *NGINX* in the search box, and then click Search.
    - Alibaba Cloud Container Registry: On the **Alibaba Cloud Container Registry** tab, you can select an image from Container Registry. You must specify the region and Container Registry instance of the image. For more information about Container Registry, see What is Container Registry?.

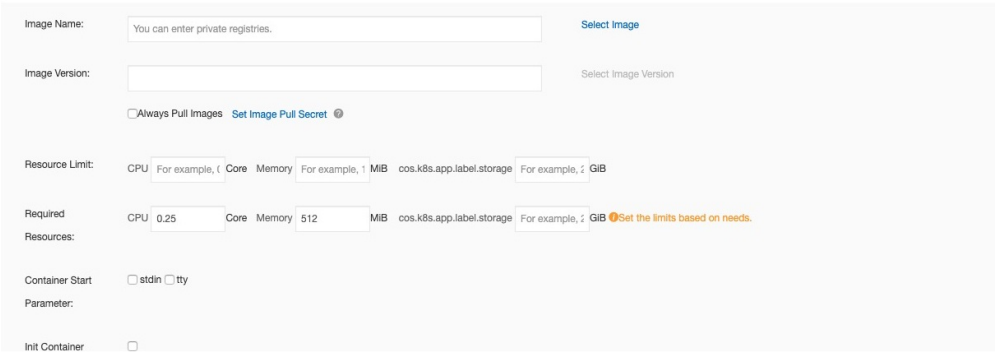
      > **Note** On the **Alibaba Cloud Container Registry** tab, you can search images by name in Container Registry.

    - Docker Official Images: On the **Docker Official Images** tab, you can select a Docker image.
    - Favorite Images: On the **Favorite Images** tab, you can select a Docker image that you have specified as one of your favorite images.
    - Search: On the **Search** tab, you can select Alibaba Cloud Images from the drop-down list and specify a region to search for an image in Container Registry. You can also select Docker Images from the drop-down list and search for a Docker image.
  - You can also enter the address of a private registry. The registry address must be in the following format: `domainname/namespace/imagename:tag`.

- **Image Version**:
  - Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used.
  - You can select the following image pull policies:
    - **ifNotPresent**: If the image that you want to pull is found on your on-premises machine, the image on your on-premises machine is used. Otherwise, ACK pulls the image from the corresponding repository.
    - **Always**: ACK pulls the image from Container Registry each time the application is deployed or scaled out.
    - **Never**: ACK uses only images on your on-premises machine.

      > **Note** If you select **Image Pull Policy**, no image pull policy is applied.

  - To pull the image without a password, click **Set Image Pull Secret** to set a Secret that is used to pull the image. For more information, see Use aliyun-acr-credential-helper to pull images without a password.
- **Always**: If you do not select this option, ACK caches the pulled image. This improves the efficiency of pulling and deploying images. If the specified image version is the same as the cached image version, ACK deploys the application from the cached image. Therefore, when you update the application code, if you do not change the image version for reasons such as to support the upper-layer workloads, the previously cached image is used. If you set Image Pull Policy to Always, ACK pulls the image from the repository each time the application is deployed. This ensures that the latest image and code are used.
- **Set Image Pull Secret**: If you use a private image, we recommend that you configure a Secret to secure the image. For more information, see Create an application by using an image pull Secret.
- **Resource Limit**: You can specify an upper limit for the CPU and memory resources that the application can consume. This prevents the application from occupying an excessive amount of resources. CPU resources are measured in millicores. Each millicore is one thousandth of one core. Memory resources are measured in bytes, which can be GiB, MiB, or KiB.
- **Required Resources**: The amount of CPU and memory resources that are reserved for the application. These resources are exclusive to the container. This prevents the application from becoming unavailable when other Services or processes occupy these resources.
- **Container Start Parameter**: Select **stdin** to send the input from the ACK console to the container. Select **tty** to send start parameters that are defined in a virtual terminal to the console.
- **Privileged Container**: If you select Privileged Container, privileged=true is set for the container. This indicates that the privilege mode is enabled. If you do not select Privileged Container, privileged=false is set for the container. This indicates that the privilege mode is disabled.
- **Init Container**: If you select this check box, an init container is created. Init containers contain practical tools. For more information, see Init Container.
- **Port**: Set the host port and container port. TCP and UDP are supported.

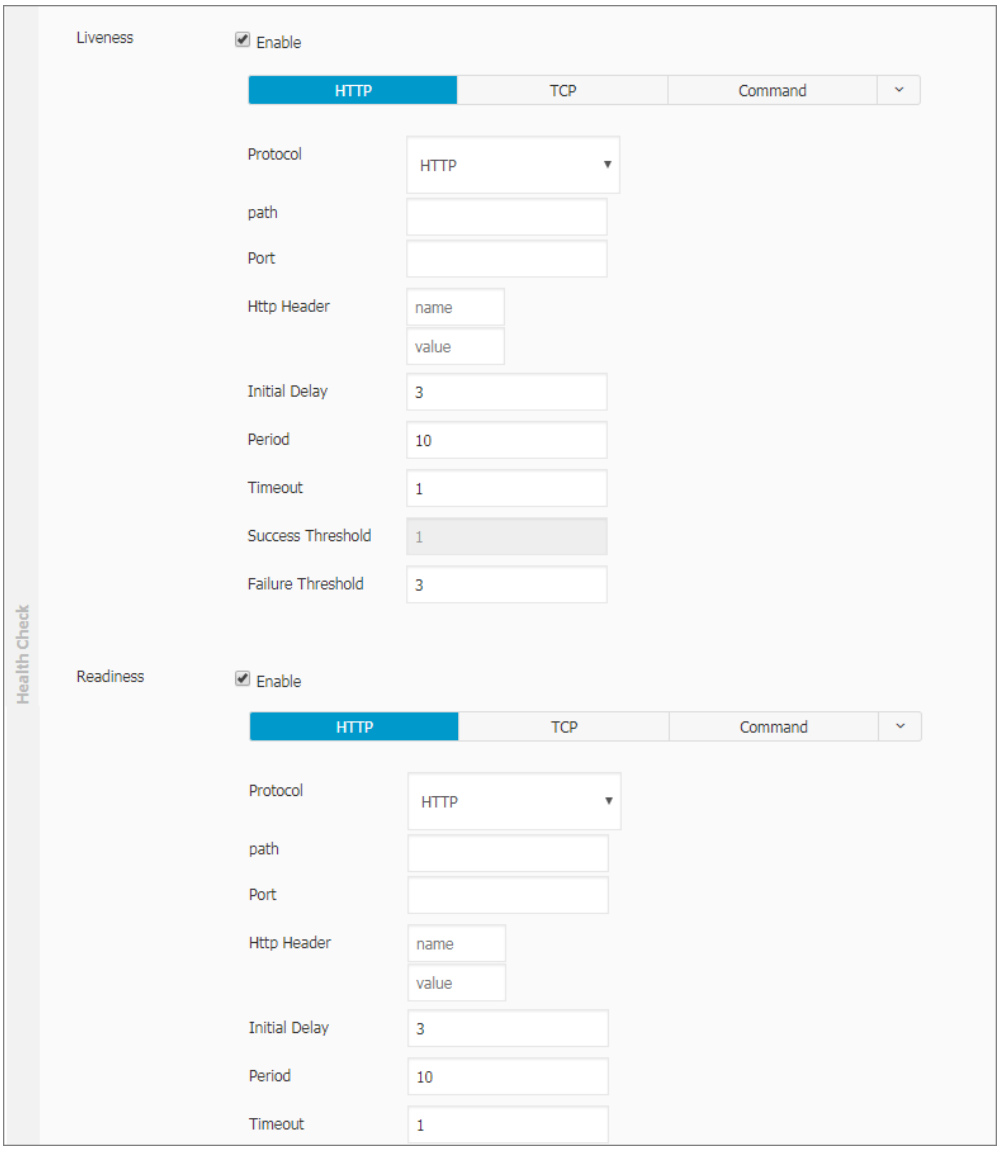


ii. (Optional)Set parameters in the **Environments** section.

You can set environment variables in key-value pairs for pods. Environment variables are used to apply pod configurations to containers. For more information, see Pod variables.

iii. (Optional)Set parameters in the **Health Check** section.

Health check settings include liveness and readiness probes. Liveness probes determine when to restart the container. Readiness probes indicate whether the container is ready to accept network traffic. For more information, see Health checks.



Request type	Description
HTTP	Sends an HTTP GET request to the container. You can set the following parameters: ■ Protocol: HTTP or HTTPS. ■ Path: the requested path on the server. ■ Port: the container port that you want to open. Enter a port number from 1 to 65535. ■ HTTP Header: the custom headers in the HTTP request. Duplicate headers are allowed. You can specify the HTTP headers in key-value pairs. ■ Initial Delay (s): the **initialDelaySeconds** field in the YAML file. This field specifies the amount of time (in seconds) that the system must wait before it can send the first probe to a launched container. Default value: 3. ■ Period (s): the **periodSeconds** field in the YAML file. This field specifies the interval (in seconds) at which probes are sent. Default value: 10. Minimum value: 1. ■ Timeout (s): the **timeoutSeconds** field in the YAML file. This field specifies the timeout period (in seconds) of probes. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

Request type	Description
TCP	Sends a TCP socket to the container. Kubelet attempts to open the socket on the specified port. If the connection can be established, the container is considered healthy. Otherwise, the container is considered unhealthy. Supported parameters include: ■ Port: the container port that you want to open. Enter a port number from 1 to 65535. ■ Initial Delay (s): the **initialDelaySeconds** field in the YAML file. This field specifies the amount of time (in seconds) that the system must wait before it can send the first probe to a launched container. Default value: 15. ■ Period (s): the **periodSeconds** field in the YAML file. This field specifies the interval (in seconds) at which probes are sent. Default value: 10. Minimum value: 1. ■ Timeout (s): the **timeoutSeconds** field in the YAML file. This field specifies the timeout period (in seconds) of probes. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

Request type	Description
Command	Runs a probe command in the container to check the health status of the container. Supported parameters include: ■ Command: the probe command that is run to check the health status of the container. ■ Initial Delay (s): the **initialDelaySeconds** field in the YAML file. This field specifies the amount of time (in seconds) that the system must wait before it can send the first probe to a launched container. Default value: 5. ■ Period (s): the **periodSeconds** field in the YAML file. This field specifies the interval (in seconds) at which probes are sent. Default value: 10. Minimum value: 1. ■ Timeout (s): the **timeoutSeconds** field in the YAML file. This field specifies the timeout period (in seconds) of probes. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

iv. (Optional)Configure the lifecycle of the container.

You can set the following parameters to configure the lifecycle of the container: Start, Post Start, and Pre Stop. For more information, see Configure the lifecycle of a container.

- **Start**: Set the command and parameter that take effect before the container starts.
- **Post Start**: Set the command that takes effect after the container starts.
- **Pre Stop**: Set the command that takes effect before the container stops.

v. (Optional)Configure volumes.

Local storage and cloud storage are supported.

- **Local storage**: You can select HostPath, ConfigMap, Secret, and EmptyDir. The storage volume is mounted to a path in the container. For more information, see Volumes.
- **Cloud storage**: supports disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets.

vi. (Optional)Configure **Log Service**. You can specify collection configurations and custom tags.

> **Note** Make sure that the Log Service agent is installed in the cluster.

You can set the Collection Configuration parameter.

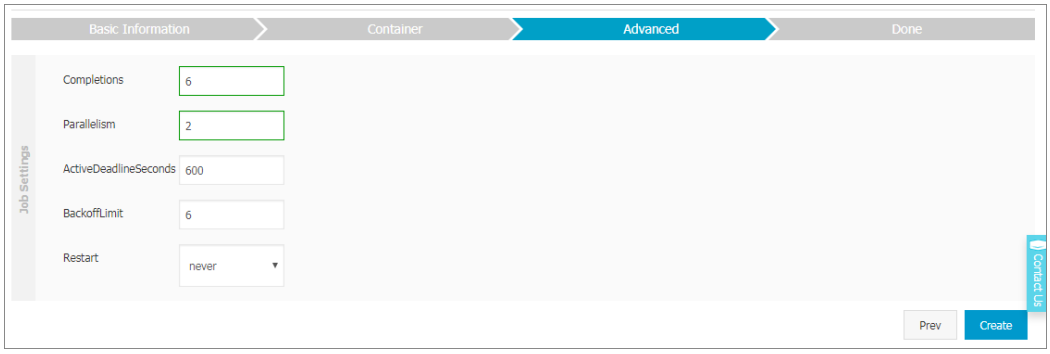
- **Logstore**: Create a Logstore in Log Service to store the collected log data.
- **Log Path in Container**: Specify stdout or a path to collect log data.
  - **stdout**: specifies that the stdout files are collected.
  - **Text Logs**: specifies that the log files in the specified path of the container are collected. Wildcard characters can be used in the path.

You can also add tags. Tags are added to the logs of the container when the logs are collected. Log data with tags is easier to aggregate and filter.

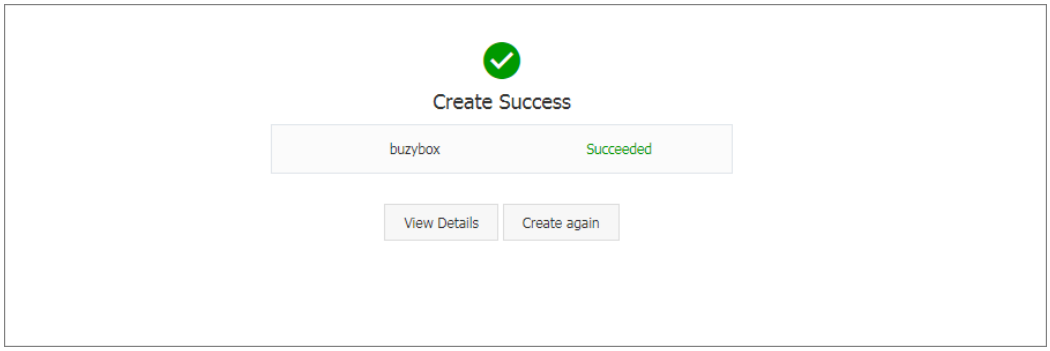
7. After you complete the configurations, click **Next**.
8. Configure advanced settings.

   You can set parameters in the **Job Setting** section.

Parameter	Description
Completions	The number of pods that the Job must run. Default value: 1.
Parallelism	The number of pods that the Job must run in parallel at any time. Default value: 1.
Timeout	The activeDeadlineSeconds field in the YAML file. This field specifies the operating time limit of the Job. If the Job is not complete within the time limit, the system attempts to terminate the Job.
BackoffLimit	The backoffLimit field in the YAML file. This field specifies the number of retries that are performed by the Job upon failure. Each time the Job fails, the failed pods associated with the Job are restarted with time delay. The time delay exponentially increases until a maximum of six minutes. Default value: 6.
Restart	Only Never and OnFailure restart policies are supported.



9. Click **Create**.
10. After the application is deployed, you are redirected to the Complete wizard page. The resource objects of the application are displayed.



You can click **View Details** to view application details.
You can view the status of pods in the **Status** column. In this example, two pods are created in parallel based on the configuration of the Job.

Wait until all pods are created.

11. Click the **Back** icon in the upper-left corner of the page. On the Jobs page, you can view the time when the Job is complete.

> **Note** If the pods associated with the Job are not complete, the end time is not displayed.

What to do next

In the left-side navigation pane of the ACK console, click **Clusters** to go to the Clusters page. Find the cluster that you want to manage, and click the cluster name or click **Details** in the **Actions** column. On the details page of the cluster, choose **Workloads** and Jobs in the left-side navigation pane. On the **Jobs** page, find the application that you want to manage, and click the application name or click **Details** in the **Actions** column. On the details page of the application, you can **scale** and **refresh** the application. You can also **view the YAML file** of the application.

- Scale: On the details page of the application, click **Scale** in the upper-right corner of the page to scale the application to a required number of replicated pods.
- View the YAML file: On the details page of the application, click **View in YAML** in the upper-right corner of the page. You can **update** and **download** the YAML file. You can also **save** the *YAML* file as a template.
- Refresh: On the details page of the application, click **Refresh** in the upper-right corner of the page to update the application details.

7.1.5. Create a CronJob

CronJobs are used to process periodic and recurring tasks. For example, you can create CronJobs to create backups or send emails. Jobs are used to process short-lived, one-off tasks. A CronJob creates one or more Jobs based on a specific schedule. This topic describes how to create a CronJob.

Create a CronJob in the ACK console

Create a CronJob from an image

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > CronJobs**.
5. In the upper-right corner of the **CronJobs** page, click **Create from Image**.
6. Set parameters for the CronJob.
   i. On the **Basic Information** wizard page, configure the basic settings. For more information, see Step 5: Configure basic settings.
   ii. On the **Container** wizard page, configure one or more containers. For more information, see Container configurations.
   iii. On the **Advanced** wizard page, configure the advanced settings.

Section	Parameter	Description
CronJobs	Schedule	You can specify a schedule on a daily, weekly, or monthly basis. You can also specify a schedule by using cron expressions. A cron expression is a string that consists of six or seven fields. These fields are separated with space characters and describe individual details of the schedule. Cron expressions support the following formats: **Seconds Minutes Hours DayofMonth Month DayofWeek Year** **Seconds Minutes Hours DayofMonth Month DayofWeek** For more information, see Cron Expressions.
	Concurrency Policy	You can select one of the following concurrency polices: ■ Allow: allows Jobs to run concurrently. Concurrent Jobs compete for cluster resources. ■ Forbid: disallows Jobs to run concurrently. If a Job is not complete within the schedule, the next Job is skipped. ■ Replace: If a Job is not complete within the schedule, the Job is skipped.
	Job History	You can specify the numbers of successful and failed Jobs for which you want to retain records. If you set the parameters to 0, the system does not retain the records of Jobs.

Section	Parameter	Description
Job Settings	Completions	For more information about how to set parameters in the Job Settings section, see Job settings.
	Parallelism	
	Timeout	
	BackoffLimit	
	Restart	
Labels,Annotations	Pod Labels	You can add labels to pods in key-value pairs. **Note** The key of a label must be 1 to 253 characters in length, and can contain only letters, digits, hyphens (-), underscores (_), and periods (.).
	Pod Annotations	You can add annotations to pods in key-value pairs. **Note** The key of an annotation must be 1 to 253 characters in length, and can contain only letters, digits, hyphens (-), underscores (_), and periods (.).

7. Click **Create**.
   After the CronJob is created, you can view the CronJob on the CronJobs page.

Create a CronJob from a YAML template

1. In the upper-right corner of the **CronJobs** page, click **Create from YAML**.
2. On the **Create** page, configure the CronJob in the **Template** section.
3. Click **Create**.

Create a CronJob by using kubectl

Before you use kubectl to create a CronJob, you must download kubectl and connect to your cluster by using kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

The following table describes the key parameters that are required to create a CronJob.

Parameter	Description
.spec.schedule	Specifies the schedule of the CronJob. For more information about the schedule format, see Cron schedule.
.spec.jobTemplate	Specifies the type of Job to be run. For more information about Job types, see Job patterns.
.spec.startingDeadlineSeconds	Specifies the due time before which a Job must be run.

Parameter	Description
.spec.concurrencyPolicy	Specifies the concurrency policy. Valid values: Allow, Forbid, and Replace. • Allow: allows Jobs to run concurrently. Concurrent Jobs compete for cluster resources. • Forbid: disallows Jobs to run concurrently. If a Job is not complete within the schedule, the next Job is skipped. • Replace: If a Job is not complete within the schedule, the Job is skipped.

To demonstrate how to create a CronJob by using kubectl, a CronJob named hello is created in this example.

1. Create a *cronjob.yaml* file and copy the following content into the file:

```
apiVersion: batch/v2alpha1
kind: CronJob
metadata:
 name: hello
spec:
 schedule: "*/1 * * * *"
 jobTemplate:
  spec:
   template:
    spec:
     containers:
     - name: hello
      image: busybox
      args:
      - /bin/sh
      - -c
      - date; echo Hello from the Kubernetes cluster
     restartPolicy: OnFailure
```

2. Run the following command to create a CronJob:

```
kubectl create -f cronjob.yaml
```

If `cronjob.batch/hello created` is returned, the CronJob is created.

What to do next

After you create a CronJob, you can perform the following operations:

- On the CronJobs page, find the created CronJob. Click **Details** in the **Actions** column to view basic information about the CronJob. The information includes the Job list, events, and logs.
- On the CronJobs page, find the created CronJob. You can choose **More > View in YAML** in the **Actions** column to view the YAML file of the CronJob. You can also choose **More > Stop** to stop the CronJob or choose **More > Delete** to delete the CronJob.

7.1.6. Manage pods

Pods are the smallest deployable units in Kubernetes. A pod runs an instance of an independent application in Kubernetes. Each pod contains one or more containers that are tightly coupled. You can modify pods, view pods, and manually scale the number of pods for an application in the Container Service for Kubernetes (ACK) console.

View pods

View pod details

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Pods**.
5. Find the pod that you want to view and click **View Details** in the Actions column.

   You can view details of pods by using one of the following methods:

   - Method 1: In the left-side navigation pane of the details page, choose **Workloads > Deployments**. Find the application that you want to manage and click the name of the application. On the **Pods** tab, click the name of the pod to view details.
   - Method 2: In the left-side navigation pane of the details page, choose **Services and Ingresses > Services**. Click the name of the Service that you want to manage. On the page that appears, find and click the name of the application that you want to manage. On the **Pods** tab, click the name of the pod to view details.

> **Note** On the Pods page, you can modify and delete pods. For pods that are created by using a Deployment, we recommend that you use the Deployment to manage the pods.

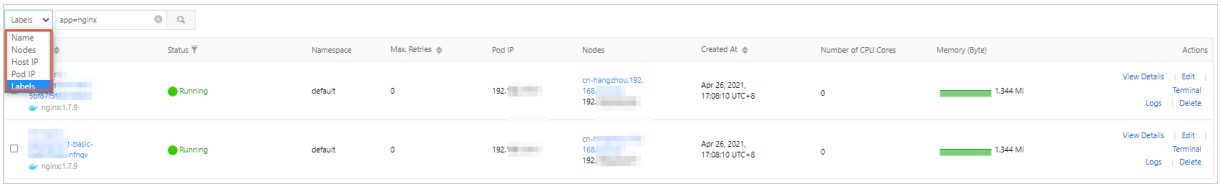
View pod logs

You can view a pod log by using the following methods:

Navigate to the Pods page, find the pod that you want to manage, and then click **Logs** on the right side to view the log data.

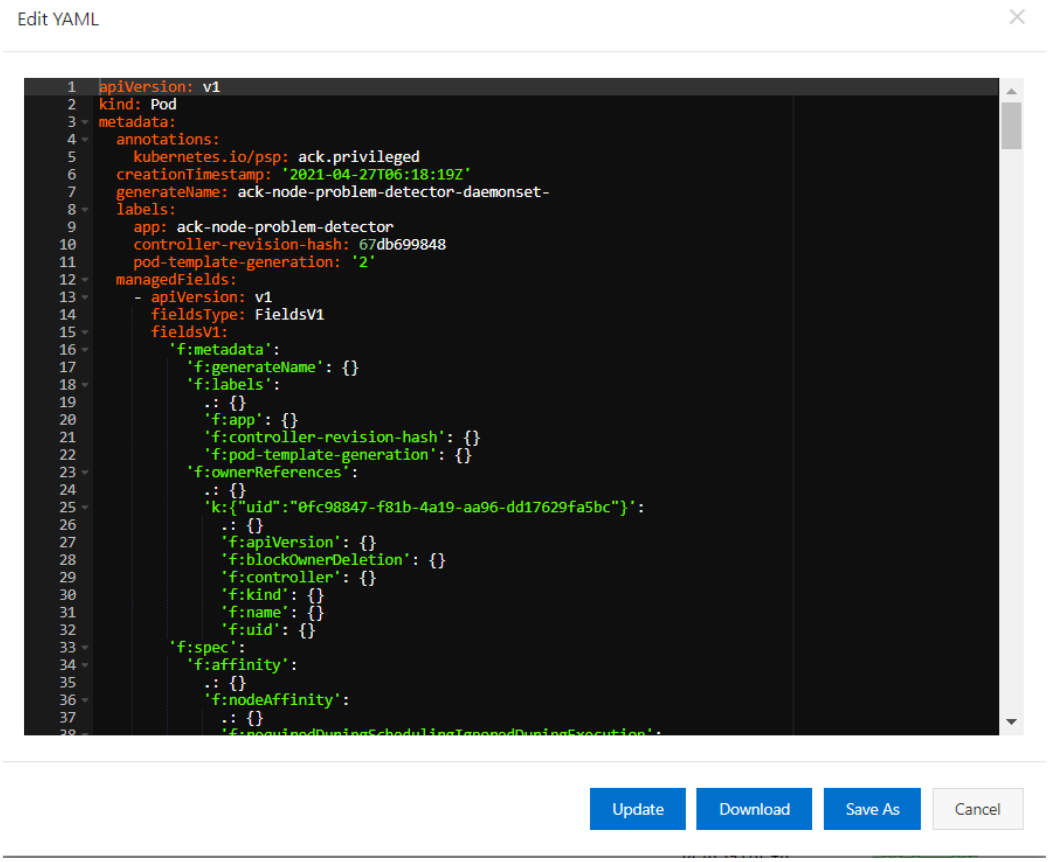
Filter pods

On the Pods page, you can filter pods by name, node, host IP, pod IP, and label.



Modify pod configurations

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Pods**.
5. On the **Pods** page, find the pod that you want to manage and click **Edit** in the Actions column.
6. In the dialog box that appears, modify the configuration of the pod and click **Update**.



Manually scale the number of pods for an application

After an application is created, you can scale the number of pods that are provisioned for the application.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. Select the namespace where the Deployment is deployed, find the Deployment, and then click **Scale** in the **Actions** column.
6. In the dialog box that appears, set Desired Number of Pods to 4 and click **OK**.

> **Note** By default, the resources created by a Deployment are updated based on the rollingUpdate strategy. This ensures that the minimum number of pods are available during the update. You can specify the minimum number of pods that must run in Pod Template of the YAML file.

7.1.7. Manage custom resources

Container Service for Kubernetes (ACK) clusters allow you to extend the Kubernetes API by adding custom resources. You can use custom resource definitions (CRDs) to define custom resources. You can view all API groups and resource types in your cluster. You can also view and manage resource objects of each resource type. This topic describes how to manage custom resources.

Limits

For some resource types, only operations such as create are supported. The resource objects of these resource types are not listed.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Custom Resources**.
5. Manage resource objects.
   - View API groups and resource types.
     On the Resource Objects tab, all API groups supported by the cluster are listed on the left side of the page. You can click the name of an API group to view all resource types that are contained in the group. You can also enter a keyword in the search box to search for resource types.
   - View resource objects.
     You can select a resource type in an API group to view the resource objects of the resource type. You can modify the YAML files of resource objects and delete resource objects.

7.2. Image

7.2.1. kritis-validation-hook introduction

kritis-validation-hook is a key component that is used to verify image signatures. You can use signature verification to ensure that only images signed by trusted authorities are deployed. This reduces the risk of malicious code execution. This topic provides examples on how kritis-validation-hook is used to verify signatures.

Context

Based on the open source project kritis, kritis-validation-hook is integrated with Alibaba Cloud Container Registry (ACR)to verify the signatures of images that are signed by Key Management Service (KMS). kritis-validation-hook is integrated with Security Center, KMS, and ACR to implement automated image signing and signature verification. This allows you to build a secure environment for clusters. For more information about how to enable signature verification for container images, see Use kritis-validation-hook to automatically verify the signatures of container images.

Permissions

To use kritis-validation-hook in a managed Kubernetes cluster, make sure that the Resource Access Management (RAM) role of worker nodes in the cluster is granted the following permissions:

```
cr:ListInstance
cr:ListMetadataOccurrences
```

If you want to grant the RAM role the preceding permissions, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the **Cluster Information** page, click the **Cluster Resources** tab and find the Worker RAM Role field.
5. Click the hyperlink in the field to go to the **RAM Roles** page in the RAM console.

6. On the **Permissions** tab, click the hyperlink in the **Policy** column to go to the **Policy Document** tab of the policy.
7. Click **Modify Policy Document**. In the Modify Policy Document panel, add the following Action policy, and click **OK** to save the modified policy.

```
{
  "Action": [
    "cr:ListInstance",
    "cr:ListMetadataOccurrences"
  ],
  "Resource": [
    "*"
  ],
  "Effect": "Allow"
}
```

Examples

The following example demonstrates how kritis-validation-hook is used to enable image signature verification for the default namespace.

Note This example does not include further details of image signing because this procedure does not involve kritis-validation-hook. The following signed image and KMS key are used in this example:

- The address of the image that is signed by KMS is kritis-demo-registry.cn-hangzhou.cr.aliyuncs.com/kritis-demo/alpine@sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45.
- The public key of the KMS key are stored in the publickey.txt file.
- The ID of the KMS key is 4a2ef103-5aa3-4220-89ee-kms-key-id.

1. Create an AttestationAuthority object to declare a trusted authority.

   The preceding public key is used in the following code block:

```
cat <<EOF > AttestationAuthority.yaml
apiVersion: kritis.grafeas.io/v1beta1
kind: AttestationAuthority
metadata:
 name: demo-aa
spec:
 noteReference: namespaces/demo-aa
 publicKeyData: $(cat publickey.txt | base64 | tr -d '\n')
 publicKeyId: 4a2ef103-5aa3-4220-89ee-kms-key-id
EOF
kubectl apply -f AttestationAuthority.yaml
```

2. Create a GenericAttestationPolicy object to declare the attestation policy. Then, specify the trusted authority for signature verification.

```
cat <<EOF > GenericAttestationPolicy.yaml
apiVersion: kritis.grafeas.io/v1beta1
kind: GenericAttestationPolicy
metadata:
 name: demo-gap
spec:
 attestationAuthorityNames:
 - demo-aa
EOF
kubectl apply -f GenericAttestationPolicy.yaml
```

3. Verify that images are not allowed to be deployed if they are not signed by the trusted authority.

   ```
   kubectl create deployment test-denied --image=alpine:3.11
   ```

   The following output is returned:

   ```
   Error from server: admission webhook "kritis-validation-hook-deployments.grafeas.io" denied the request: image alpine:3.11 is not attested
   ```

   Run the following command:

   ```
   kubectl create deployment test-denied --image=kritis-demo-registry.cn-hangzhou.cr.aliyuncs.com/kritis-demo/alpine:3.11
   ```

   The following output is returned:

   ```
   Error from server: admission webhook "kritis-validation-hook-deployments.grafeas.io" denied the request: image kritis-demo-registry.cn-hangzhou.cr.aliyuncs.com/kritis-demo/alpine:3.11 is not attested
   ```

4. Verify that images are allowed to be deployed if they are signed by the trusted authority.

   Run the following command:

   ```
   kubectl create deployment test-allow --image=kritis-demo-registry.cn-hangzhou.cr.aliyuncs.com/kritis-demo/alpine@sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45
   ```

   The following output is returned:

   ```
   deployment.apps/test-allow created
   ```

Configure an image verification whitelist

If the middleware or third-party component automatically injects sidecar containers, the images of the sidecar containers may fail the image signature verification. This applies to service mesh scenarios and may cause issues when you attempt to deploy pods. To fix this issue, you can use kritis-validation-hook to configure an image verification whitelist. This way, kritis-validation-hook verifies only the signatures of the images that are not included in the whitelist. The signatures of the images that are included in the whitelist are not verified.

You can define the `admissionallowlists.kritis.grafeas.io` resource to configure the image verification whitelist. This resource is defined in the following CustomResourceDefinition (CRD):

```
apiVersion: kritis.grafeas.io/v1beta1  # The default value. You do not need to modify the value.
kind: AdmissionAllowlist          # The default value. You do not need to modify the value.
metadata:
 name: kritis-allowlist          # The name of the resource. This name is unique within a cluster.
spec:
 patterns:                  # The whitelist. You can specify one or more images in the whitelist.
 - namePattern: 'registry*. *.aliyuncs.com/acs/*' # The image name that you want to add to the whitelist. For mor
e information, see the following descriptions:
 - namePattern: 'registry-vpc.cn-beijing.aliyuncs.com/arms-docker-repo/*'
  namespace: 'default'  # Optional The namespace to which the specified whitelist applies. If you do not set the v
alue, the specified whitelist applies to all namespaces.
```

To add an image of ACK to the whitelist, perform the following steps:

1. Define the whitelist in the following CRD and save the CRD as `kritis-admission-allowlist-acs.yaml`.

   ```
   apiVersion: kritis.grafeas.io/v1beta1
   kind: AdmissionAllowlist
   metadata:
    name: allow-acs-images
   spec:
    patterns:
    - namePattern: 'registry*. *.aliyuncs.com/acs/*'
   ```

   The namePattern parameter supports exact matching and wildcard matching. Asterisks (*) are used as wildcard characters.

   - If the value of namePattern does not contain asterisks (*), exact matching is used. For example, `nginx:v0.1.0` matches only `nginx:v0.1.0`.
   - If asterisks (*) are used in wildcard matching, the following requirements must be met:
     - The asterisk (*) that appears at the end of the value matches all characters except forward slashes (/). For example, `a.com/nginx*` matches `a.com/nginx:v0.1.0`, but does not match `a.com/nginx/test:v0.1.0`.
     - The asterisk (*) that does not appear at the end of the value matches letters, digits, hyphen (-), and underscores (_). For example, `registry-vpc.cn-*.aliyuncs.com/acs/pause:3.2` matches both `registry-vpc.cn-hangzhou.aliyuncs.com/acs/pause:3.2` and `registry-vpc.cn-beijing.aliyuncs.com/acs/pause:3.2`.

   The following list shows the common images that are included in the whitelist. You can add more items to the whitelist based on your business requirements.

   ```
   # The images of Container Service for Kubernetes (ACK).
   - namePattern: 'registry*. *.aliyuncs.com/acs/*'
   # The images of ACK in the regions within China.
   - namePattern: 'registry*.cn-*.aliyuncs.com/acs/*'
   # The images of Application Real-Time Monitoring Service (ARMS).
   - namePattern: 'registry*. *.aliyuncs.com/arms-docker-repo/*'
   # The images of ARMS in the regions inside China.
   - namePattern: 'registry*.cn-*.aliyuncs.com/arms-docker-repo/*'
   ```

2. To apply the defined whitelist, run the following command:

   ```
   kubectl apply -f kritis-admission-allowlist-acs.yaml
   ```

   The following output is returned:

   ```
   admissionallowlist.kritis.grafeas.io/allow-acs-images created
   ```

3. Run the `kubectl get admissionallowlists.kritis.grafeas.io` command to check the specified whitelist.

   The following output is returned:

   ```
   NAME         AGE
   allow-acs-images  2m22s
   ```

Next up

kritis-validation-hook will be integrated with other Alibaba Cloud services to provide more advanced features. The features are not limited to:

- Immutable image tags. You can use this feature to specify tags instead of image digests when you verify image signatures. This improves user experience. The latest version of kritis-validation-hook can verify the signatures of images with immutable image tags. For more information, see Configure a repository to be immutable.
- Image vulnerability detection. After you use this feature, images that contain vulnerabilities of specified levels are not allowed to be deployed. This ensures the security of your environment.

Related information

- kritis-validation-hook
- Use kritis-validation-hook to automatically verify the signatures of container images

7.2.2. Manage images

This topic describes how to create and view images.

Create images

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > Alibaba Cloud Container Registry**.
3. On the **Instances** page, click the instance of Container Registry Enterprise Edition that you want to manage.

   To create an instance of Container Registry Enterprise Edition, see Step 3: Create a Container Registry Enterprise Edition instance.
4. Create a namespace in the instance of Container Registry Enterprise Edition. For more information, see Step 4: Create a namespace.
5. Create a repository in the instance of Container Registry Enterprise Edition. For more information, see Step 5: Create an image repository.
6. Push an image to the repository, or automatically build an image from a third-party code repository. For more information, see Use a Container Registry Enterprise Edition instance to push and pull images and Use Container Registry Enterprise Edition to build images.

View images

1. Log on to the Container Registry console.
2. In the top navigation bar, select a region.
3. In the left-side navigation pane, click **Instances**.
4. On the **Instances** page, click the required Container Registry Enterprise Edition instance.
5. On the management page of the instance, choose **Repositories > Repositories** in the left-side navigation pane. On the Repositories page, click the name of the repository that you want to manage.
6. On the details page of the repository, click **Tags** in the left-side navigation pane. On the **Tags** page,

you can view images that are pushed to the repository.

7.2.3. Use aliyun-acr-credential-helper to pull images without a password

Container Registry provides the aliyun-acr-credential-helper component for you to pull private images without a password from Container Registry Enterprise Edition or Personal Edition. This component is automatically installed in Container Service for Kubernetes (ACK) clusters. This topic describes how to use aliyun-acr-credential-helper to pull a private image without a password in different scenarios.

Prerequisites

- 创建Kubernetes托管版集群
- Connect to Kubernetes clusters by using kubectl

Notice

- To use aliyun-acr-credential-helper, do not manually specify the imagePullSecret field. If the imagePullSecret field is specified in the template of a Kubernetes resource, such as a Deployment, the component becomes invalid.
- If a Kubernetes resource, such as a Deployment, uses custom service accounts, you must modify the service-account field in the configuration file of the component. Then, the component is authorized to pull images with the custom service accounts.
- Check whether the private image that you want to pull is in the region where your ACK cluster is deployed. By default, you can pull private images from only Container Registry instances that are deployed in the region where your ACK cluster is deployed. If you want to pull images from Container Registry instances that are deployed in different regions, see Scenario 3 in this topic.
- If you want to assume a custom Resource Access Management (RAM) role to pull private images, you must specify the AccessKey ID and AccessKey secret of the RAM role in the acr-configuration ConfigMap. However, this may disclose the AccessKey information. To ensure data security, make sure that the RAM role is granted only the permissions to pull images.
- After you create a service account in a cluster, it takes some time for aliyun-acr-credential-helper to renew the token of the service account. The new token for pulling private images is generated based on the default permissions of the ACK cluster. Applications with the service account can use the token to pull images only after the token is renewed. If you create an application immediately after you create a service account, the application will fail to pull images because it is unauthorized.
- By default, the configuration of aliyun-acr-credential-helper overwrites the imagePullSecret field of default service accounts in all namespaces. These service accounts are automatically modified when the service-account field of the **acr-configuration** ConfigMap in the kube-system namespace is changed.
- When you modify the **acr-configuration** ConfigMap in the kube-system namespace, make sure that you use the same indentation as the example in this topic. We recommend that you paste the YAML code provided in this topic to the editor, replace the corresponding values, and apply the configuration. This ensures that the format of the ConfigMap is valid.

Context

aliyun-acr-credential-helper reads the required information from the acr-configuration ConfigMap that is created in the kube-system namespace and then pulls private images. You can pull private images by using the following methods:

- Assume the worker role of an ACK cluster to pull private images from Container Registry instances that are created within your account.
- Use the AccessKey ID and AccessKey secret of a custom RAM role to pull private images from Container Registry instances that are created within your account.
- Use AssumeRole to assume the RAM role of another account to pull private images from that account.

aliyun-acr-credential-helper supports the following images and clusters:

- Supported images
  - You can use aliyun-acr-credential-helper to pull private images from instances of Container Registry Enterprise Edition and Personal Edition.
  - You can use the component to pull private images from your Container Registry instances. You can also pull private images from other accounts after authorization or by using the AccessKey ID and AccessKey secret.
  - You can use the component to pull private images from Container Registry instances that are deployed in different regions.
- Supported clusters
  - You can use the component to pull images without a password from clusters in multiple namespaces.
  - Supported cluster types:
    - Dedicated Kubernetes clusters.
    - Managed Kubernetes clusters.
  - Supported cluster versions:
    - Dedicated Kubernetes clusters: The Kubernetes version must be V1.11.2 or later. If the Kubernetes version is earlier than V1.11.2, you must manually upgrade Kubernetes to V1.11.2 or later. For more information, see Upgrade a cluster.
    - Managed Kubernetes clusters: All versions.

Upgrade and configure the component

Before you use the aliyun-acr-credential-helper component to pull images, you must perform the following steps:

1. Upgrade the aliyun-acr-credential-helper component.
   i. Log on to the the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and choose **More > Manage System Components** in the Actions column.
   iv. On the page that appears, find **aliyun-acr-credential-helper** in the **Security** section and click **Upgrade**.
2. Configure acr-configuration.

   Configure acr-configuration in the ACK console.
   i. Log on to the the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the Actions column.
   iv. In the left-side navigation pane of the **details** page, choose **Configurations > ConfigMaps**.
   v. At the top of the **ConfigMap** page, select **kube-system** from the Namespace drop-down list. Then, find and configure **acr-configuration** by using one of the following methods:

- Method 1: Click **Edit** on the right side of acr-configuration, and enter keys and values in the **ConfigMap**.
  If **acr-configuration** is not found in the list of ConfigMaps, see Create a ConfigMap. For more information about how to update a ConfigMap, see Modify a ConfigMap.
- Method 2: Click **Edit YAML** on the right side of acr-configuration, and enter keys and values in the **ConfigMap**.

The following table describes the keys and values of the acr-configuration ConfigMap.

Key	Description	Value
service-account	The service accounts that are used by the component to pull images.	Default value: **Default**. **Note** Separate multiple service accounts with commas (,). Enter an asterisk (*) to specify all service accounts in the specified namespace.

acr-registry-info	The information about Container Registry instances. Each instance can be specified by three string type fields in a YAML file.
	Note Set the three fields based on the following descriptions:
	■ instanceId: the ID of the Container Registry instance. This field is required for instances of Container Registry Enterprise Edition.
	■ regionId: the ID of the region where the Container Registry instance is deployed. This field is optional. The default value is the region where your ACK cluster is deployed.
	■ domains: the domain names of the Container Registry instance. This field is optional. By default, all domain names of the instance are specified. Separate multiple domain names with commas (,).
		By default, this parameter is not specified. This means that images are pulled from the default repository of the Container Registry instance that is deployed in the region where the ACK cluster is deployed.
		The following template shows the configuration of an instance of Container Registry Enterprise Edition:
		`- instanceId: cri-xxx`
		`  regionId: cn-hangzhou`
		`  domains: xxx.com,yyy.com`
		The following template shows the configuration of an instance of Container Registry Personal Edition:
		`- instanceId: ""`
		`  regionId: cn-hangzhou`
		`  domains: xxx.com,yyy.com`

Key	Description	Value
watch-namespace	The namespaces to which the images to be pulled belong.	Default value: **Default**. **Note** If the value is set to all, images are pulled from all namespaces without a password. Separate multiple namespaces with commas (,).
expiring-threshold	The duration after which the cache token expires.	Default value: **15m**. We recommend that you use the default value.

Configure acr-configuration by using kubectl

i. Run the following command to open the ConfigMap of acr-configuration:

```
kubectl edit cm acr-configuration -n kube-system
```

ii. Configure the parameters of acr-configuration based on your requirements.

The following templates show the configurations of acr-configuration for instances of Container Registry Enterprise Edition and Personal Edition:

- Enterprise Edition

```
apiVersion: v1
data:
 acr-api-version: "2018-12-01"
 acr-registry-info: |-
  - instanceId: "cri-xxx"
    regionId: "cn-hangzhou"
 expiring-threshold: 15m
 service-account: default
 watch-namespace: all
kind: ConfigMap
metadata:
 name: acr-configuration
 namespace: kube-system
 selfLink: /api/v1/namespaces/kube-system/configmaps/acr-configuration
```

- Personal Edition

```
apiVersion: v1
data:
 acr-api-version: "2018-12-01"
 acr-registry-info: |-
  - instanceId: ""
    regionId: "cn-hangzhou"
 expiring-threshold: 15m
 service-account: default
 watch-namespace: all
kind: ConfigMap
metadata:
 name: acr-configuration
 namespace: kube-system
 selfLink: /api/v1/namespaces/kube-system/configmaps/acr-configuration
```

Scenario 1: Pull private images from instances of Container Registry Enterprise Edition and Personal Edition

ACK allows you to pull private images from Container Registry Enterprise Edition, Container Registry Personal Edition, or both Container Registry Enterprise Edition and Personal Edition. Modify the **acr-configuration** ConfigMap to meet your requirements. For more information about how to configure the component, see Configure aliyun-acr-credential-helper. Sample configurations:

- Pull private images from Container Registry Enterprise Edition.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
   - instanceId: "cri-xxx"
     regionId: "cn-hangzhou"
     domains: "xxx.com","yyy.com"
```

- Pull private images from Container Registry Personal Edition.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
   - instanceId: ""
     regionId: "cn-hangzhou"
     domains: "xxx.com","yyy.com"
```

- Pull private images from both Container Registry Enterprise Edition and Personal Edition.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
   - instanceId: ""
   - instanceId: "cri-xxxx"
```

Scenario 2: Pull images with the current account

If you want to pull images with the current account, you must check whether the current account has the permissions to pull images from Container Registry instances.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the **details** page of the cluster, click the **Cluster Resources** tab. On the Cluster Resources tab, click the hyperlink next to **Worker RAM role**.
5. On the page that appears, click the **Permissions** tab, and click the name of the policy that you want to manage in the **Policy** column.
6. Click **Modify Policy Document**.

7. In the **Policy Document** section, add the following content and click **OK**:

```
{
  "Action": [
    "cr:Get*",
    "cr:List*",
    "cr:PullRepository"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
```

Note If you fail to pull images without a password with the current account, you must check whether the image specified in the pod exists in the Container Registry repository.

- The image exists in the Container Registry repository but is not found on the Container Registry instances that are created by the current account. In this case, you must pull the image across accounts. For more information, see Scenario 4.
- The image exists in the Container Registry repository and is found on the Container Registry instances that are created by the current account. In this case, Submit a ticket.
- The image is not found in the Container Registry repository. In this case, you must check whether the image can be found on Container Registry. If the image is not found on Container Registry, you must upload the image to Container Registry or make the image publicly available over the Internet.

Scenario 3: Pull images across regions

If you want to pull images from Container Registry instances that are deployed in different regions, you must modify the **acr-configuration** ConfigMap.

For example, you want to pull images from Container Registry instances that are deployed in the China (Beijing) and China (Hangzhou) regions at a time. In this case, modify acr-configuration as shown in the following code block. For more information about how to configure the component, see Configure aliyun-acr-credential-helper.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
    - instanceId: ""
      regionId: cn-beijing
    - instanceId: ""
      regionId: cn-hangzhou
```

Scenario 4: Pull images across accounts

In this scenario, you can pull images from a different account by using the following methods:

- **Pull images from a different account by role assuming**: Account A assumes the RAM role of Account B to pull private images from Account B.
- **Pull images from a different account by using the AccessKey pair of the account**: The account whose RAM role you want to assume must have the permissions to pull images.

Pull images from a different account by role assuming

Note
Follow these rules when you pull images from a different account:

1. The RAM role of Account B is authorized to pull private images from a specified private repository. This rule requires you to grant the **cr.*** permission (full permissions on Container Registry) to the RAM role of Account B.
2. Account B allows the worker role of the ACK cluster created by Account A to assume the RAM role of Account B. This rule requires you to modify the trust policy of the RAM role of Account B.
3. The worker role of the ACK cluster created by Account A has the permissions to assume the RAM role of Account B. This rule requires you to attach the AliyunAssumeRoleAccess permission policy to the worker role of Account A.
4. Set the worker role of Account A to assume the RAM role of Account B. This rule requires you to specify the assumeRoleARN field in the acr-configuration ConfigMap.

1. Create a RAM role of Account B. Specify **Alibaba Cloud accounts** as the trusted entity of the RAM role. Make sure that the created RAM role has the permissions to pull private images from Account B.
   i. Create a RAM role.

      For more information, see Create a RAM role for a trusted Alibaba Cloud account.
   ii. Customize the permissions of the created RAM role and make sure that the created RAM role has the permissions to pull private images from Account B.

      For more information, see Modify a custom policy.

      Notice Make sure that you have granted the **cr.*** permission to the RAM role of Account B.

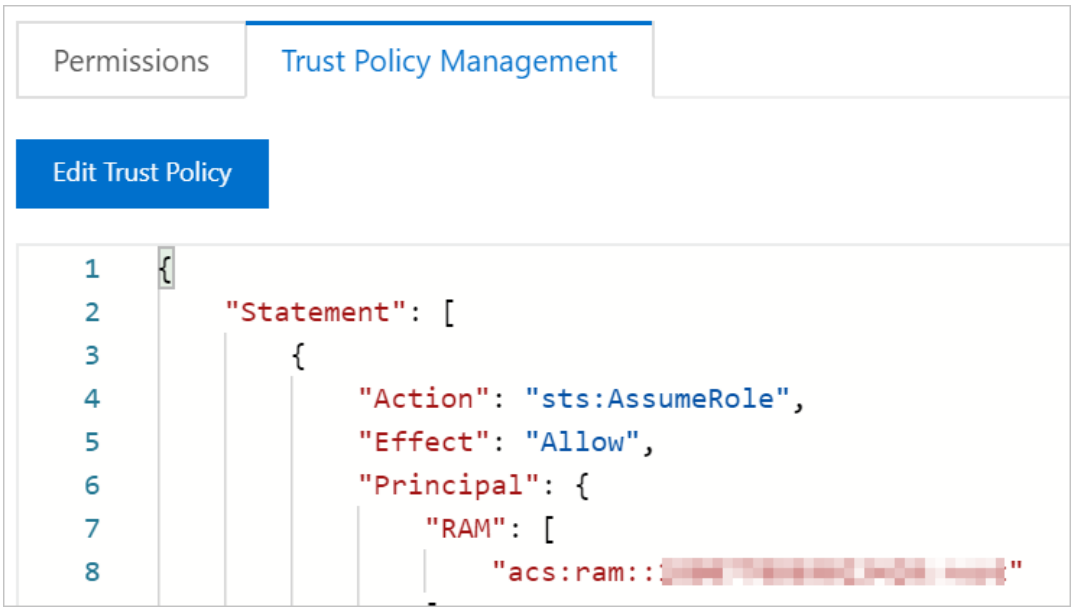
      ```
      {
      "Action": [
       "cr:GetAuthorizationToken",
       "cr:ListInstanceEndpoint",
       "cr:PullRepository"
      ],
      "Resource": [
      "*"
      ],
      "Effect": "Allow"
      }
      ```

      The following figure shows where the content is added.

2. Modify the trust policy of the RAM role of Account B. This way, Account B allows the worker role of the ACK cluster created by Account A to assume the RAM role of Account B.
   i. Obtain the Alibaba Cloud Resource Name (ARN) of the worker role of the ACK cluster created by Account A.

      For more information, see How do I find the ARN of a RAM role?
   ii. Modify the trust policy of the RAM role of Account B.
      a. Log on to the RAM console. In the left-side navigation pane, choose **RAM Roles**. On the RAM Roles page, find and click the RAM role of Account B.
      b. On the details page of the RAM role of Account B, click the **Trust Policy Management** tab, and enter the ARN of the worker role into the Principal field of the trust policy.



3. Check whether the worker role of Account A has the AssumeRole permission.

Applicable Scope of Permission	Policy
All	AliyunSTSAssumeRoleAccess

   For more information, see View the basic information about a policy.

4. Add the assumeRoleARN field to the ConfigMap of the aliyun-acr-credential-helper component.

   Set the value of the assumeRoleARN field to the ARN of the RAM role for Account B. For more information about how to obtain the ARN information, see How do I find the ARN of a RAM role? The following YAML file is used as an example. For more information about how to configure the component, see the preceding Configure aliyun-acr-credential-helper section.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
   - instanceId: ""
     regionId: cn-beijing
     domains: registry.cn-beijing.aliyuncs.com
     assumeRoleARN: acs:ram::.*:role/kubernetesworkerrole-test
```

Pull images from a different account by using the AccessKey pair of the account

1. Create a RAM role for your account and grant the RAM role the permissions to pull images from Container Registry.

   For more information about how to authorize an account to pull images from Container Registry, see Step 1 in Pull images from a different account by role assuming.

2. Configure the acr-configuration ConfigMap in the **kube-system** namespace. Specify the AccessKey ID and AccessKey secret of the created RAM role.

   This way, aliyun-acr-credential-helper can assume the RAM role to pull private images. For more information about how to obtain the AccessKey information, see View the basic information about AccessKey pairs.
   The following code block provides an example. For more information about how to configure the component, see Configure aliyun-acr-credential-helper.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
   - instanceId: ""
     customAccessKey: "xxxxx" // Enter the AccessKey ID of the created RAM user.
     customAccessKeySecret: "xxxxxx" // Enter the AccessKey secret of the created RAM user.
```

7.2.4. Use kritis-validation-hook to automatically verify the signatures of container images

This topic describes how to use Container Registry, Key Management Service (KMS), Security Center, and kritis-validation-hook to automatically verify the signatures of container images. This allows you to deploy only container images that have been signed by trusted authorities. This also reduces the risk of malicious code execution in your environment.

Step 1: Install kritis-validation-hook

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Operations > Add-ons**.
5. In the **Optional Add-ons** section of the **Add-ons** page, find **kritis-validation-hook** and click **Install** in the Actions column of the add-on.

6. (Optional)Authorize your cluster to run kritis-validation-hook.

   If you want to run kritis-validation-hook in a managed Kubernetes cluster, you must grant the cluster the permissions to access related resources. For more information, see Permissions.

   > **Note** By default, dedicated Kubernetes clusters are granted the permissions to run kritis-validation-hook.

Step 2: Create a key that is used to sign images in the KMS console

1. Log on to the KMS console.
2. In the upper-left corner of the **Keys** page, click **Create Key**.
3. In the **Create Key** dialog box, set **Alias Name** and **Description**.

   > **Notice** Specify **RSA_2048** as **Key Spec** and specify **SIGN/VERIFY** as **Purpose**.

4. Click **Advanced** and specify **Key Material Source**.
   - **Alibaba Cloud KMS**: Use KMS to generate key material.
   - **External**: Import key material from an external source. For more information about how to import key material, see Import key material.

     > **Note** If you select External, you must also select **I understand the implications of using the external key materials key**.

5. Click **OK**.

Step 3: Create a witness that uses the key in the Security Center console

1. Log on to the Security Center console.
2. In the left-side navigation pane, choose **Operation > Container Signature**.
3. On the **Witness** tab, click **Create Witness**.

   Set the witness information. For more information, see Use the container signature feature.

4. Click **OK**.

Step 4: Enable image signing in Container Registry

1. Log on to the Container Registry console.
2. On the **Instances** page, find the Enterprise Edition instance that you want to manage and click the instance name or click **Manage** in the Actions column.
3. In the left-side navigation pane of the Manage page, choose **Repositories > Namespaces**.

   Create a namespace and enable image signing for images in the namespace. For more information, see Manage namespaces.

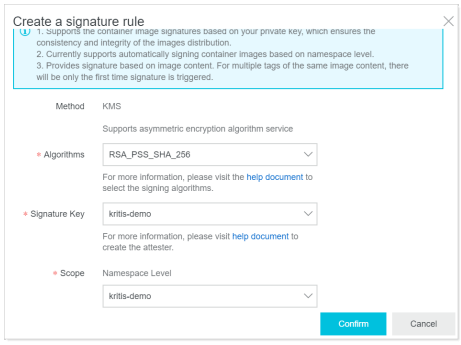
4. Enable image signing for the created namespace.

   When you create signature signing rules, select the witness that is created in Step 3.

   i. In the left-side navigation pane, choose **Content Trust > Image Signature**.

ii. On the **Image Signature** page, click **Create Signature Rule**.

For more information about how to configure signature signing rules, see Configure a signature rule for automatic image signing.

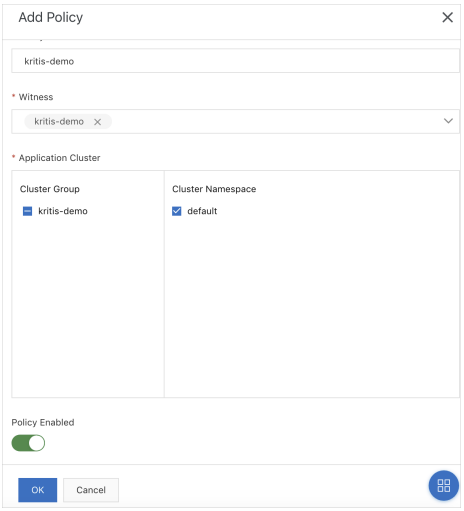


Step 5: Enable signature verification in Security Center

To enable signature verification for a namespace, add and enable a security policy for the cluster in the Security Center console.

1. Log on to the Security Center console.
2. In the left-side navigation pane, choose **Operation > Container Signature**.
3. On the **Security Policy** tab, click **Add Policy**.

   For more information about how to add a security policy, see Create a security policy.



4. Click **OK**.

Step 6: Check whether signature verification is enabled

> **Note** Only images in digest format are supported.

Run the following command to check whether signature verification is enabled:

- After signature verification is enabled for the default namespace, unsigned images cannot be deployed in the namespace.
  To specify an image by tag, run the following command:

```
kubectl -n default create deployment not-sign --image=alpine:3.11 -- sleep 10
```

```
Error from server: admission webhook "kritis-validation-hook-deployments.grafeas.io" denied the request: im
age alpine:3.11 is not attested
```

To specify an image by digest, run the following command:

```
kubectl -n default create deployment not-sign --image=alpine@sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1
f23a8e742c90e6e1297bfa1bc0c45 -- sleep 10
```

```
Error from server: admission webhook "kritis-validation-hook-deployments.grafeas.io" denied the request: im
age alpine@sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45 is not attested
```

- Push an image to a namespace where image signature signing is enabled, and verify that the image can be deployed after the image is signed.

```
docker push kritis-demo-registry-vpc.cn-hongkong.cr.aliyuncs.com/kritis-demo/alpine:3.11
```

```
The push refers to repository [kritis-demo-registry-vpc.cn-hongkong.cr.aliyuncs.com/kritis-demo/alpine]
5216338b40a7: Pushed
3.11: digest: sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45 size: 528
```

To accept requests to deploy signed images, run the following command:

```
kubectl -n default create deployment is-signed --image=kritis-demo-registry-vpc.cn-hongkong.cr.aliyuncs.com/
kritis-demo/alpine@sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45 -- sleep 1
0
```

```
deployment.apps/is-signed created
```

Related information

References

For more information about kritis-validation-hook, see kritis-validation-hook introduction.
For more information about the release notes for kritis-validation-hook, see kritis-validation-hook.

7.3. Configuration items and key

7.3.1. Manage ConfigMaps

In the Container Service for Kubernetes (ACK) console, you can create a ConfigMap on the ConfigMap page or from a YAML template. You can use ConfigMaps to store non-sensitive, unencrypted configuration information. This topic describes how to manage ConfigMaps.

Prerequisites

An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.

Create a ConfigMap

Create a ConfigMap on the ConfigMap page

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
5. On the **ConfigMap** page, select the namespace to which the ConfigMap belongs from the **Namespace** drop-down list, and click **Create** in the upper-right corner.

6. In the Create panel, enter a name for the ConfigMap, click **+ Add**, enter **keys** and **values**, and then click **OK**.
   - **ConfigMap Name**: Required. The name of the ConfigMap. The name can contain lowercase letters, digits, hyphens (-), and periods (.). Other resource objects must reference the ConfigMap name to obtain configuration information.
   - **ConfigMap**: Enter the names and values of the ConfigMap in key-value pairs. You can also click **Import** to create the ConfigMap from a file.
7. Click **OK**.
   You can find the ConfigMap named aliyun-config on the ConfigMap page.

Create a ConfigMap by using a YAML template

1. Log on to the ACK console.
2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
4. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
5. On the page that appears, set the parameters and click **Create**.

Parameter	Description
Sample Template	You can select Custom from the drop-down list and configure the ConfigMap in YAML syntax. You can also select **Resource - ConfigMap** from the drop-down list. If you select Resource - ConfigMap, the ConfigMap is named aliyun-config and contains the following variable files: `game.properties` and `ui.properties`. You can modify the ConfigMap to meet your requirements.

6. Click **Create**.
   After the ConfigMap named aliyun-config is created, you can find it on the ConfigMap page.

View a ConfigMap

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
5. On the **ConfigMap** page, find the ConfigMap that you want to view and click its name.
6. On the details page of the ConfigMap, you can view details about the ConfigMap. You can also view the key-value pairs in the ConfigMap.

Modify a ConfigMap

If you modify a ConfigMap, the applications that use this ConfigMap are affected.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
5. On the **ConfigMap** page, find the ConfigMap that you want to modify and click **Edit** in the Actions column.
6. In the **Edit** panel, modify the **keys** and **values** based on your requirements and click **OK**.

> **Note** You can also modify the keys and values in the YAML file. On the **ConfigMap** page, click **Edit YAML** in the Actions column. In the **View in YAML** panel, modify the keys and values based on your requirements and click **OK**.

Delete a ConfigMap

You can delete a ConfigMap that is no longer in use.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
5. On the **ConfigMap** page, find the ConfigMap that you want to delete and click **Delete** in the Actions column.
6. In the **Confirm** message, click **OK**.

7.3.2. Configure a pod to use a ConfigMap

This topic describes how to configure a pod to use a ConfigMap.

Context

You can configure a pod to use a ConfigMap in the following scenarios:

- Use a ConfigMap to define environment variables for a pod.
- Use a ConfigMap to set command line parameters.
- Mount a ConfigMap as a volume to a pod.

For more information, see Configure a pod to use a ConfigMap.

Limits

To configure a pod to use a ConfigMap, make sure that the ConfigMap and pod are in the same cluster and namespace.

Create a ConfigMap

In this example, a ConfigMap named special-config is created. This ConfigMap consists of two key-value pairs: `SPECIAL_LEVEL: very` and `SPECIAL_TYPE: charm`.

The following YAML template is used to create the ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
  namespace: default
data:
  SPECIAL_LEVEL: very
  SPECIAL_TYPE: charm
```

Use a ConfigMap to define environment variables for a pod.

Use the key-value pairs of a ConfigMap to define environment variables for a pod

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Select the sample template or enter a custom template, and click **Create**.

   To define an environment variable for a pod, you can use the `valueFrom` field to reference the value of SPECIAL_LEVEL.
   The following code block is an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: config-pod-1
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:                ##Use the valueFrom field to denote that env references the value of a ConfigM
ap.
            configMapKeyRef:
              name: special-config          ##The name of the referenced ConfigMap.
              key: key: SPECIAL_LEVEL           ##The key of the referenced ConfigMap.
  restartPolicy: Never
```

Use all key-value pairs of a ConfigMap to define multiple environment variables for a pod

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Select the sample template or enter a custom template, and click **Create**.

To use all key-value pairs of a ConfigMap to define multiple environment variables for a pod, you can use the envFrom parameter. The keys of the ConfigMap are used as the names of the environment variables.
The following code block is an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: config-pod-2
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      envFrom:          ##Reference all key-value pairs of the sepcial-config ConfigMap.
      - configMapRef:
          name: special-config
  restartPolicy: Never
```

Use a ConfigMap to set command line parameters

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Select the sample template or enter a custom template, and click **Create**.

   You can use ConfigMaps to define the commands or parameter values for a container by using the environment variable replacement syntax `$(VAR_NAME)`.
   The following code block is an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: config-pod-3
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "echo $(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_LEVEL
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: SPECIAL_TYPE
  restartPolicy: Never
```

Mount a ConfigMap as a volume to a pod.

1. In the left-side navigation pane of the ACK console, click **Clusters**.
2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
4. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
5. Select the sample template or enter a custom template, and click **Create**.

   To mount a ConfigMap as a volume, specify the name of the ConfigMap in the volumes section. This saves the key-value pairs of the ConfigMap to the directory specified in the mountPath field. In this example, the directory is /etc/config. Configuration files that are named after the keys of the ConfigMap are generated. The values of the ConfigMap are stored in the related files.
   The following code block is an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: config-pod-4
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "ls /etc/config/"    ##List the files in the directory.
      volumeMounts:
      - name: config-volume
        mountPath: /etc/config
  volumes:
    - name: config-volume
      configMap:
        name: special-config
  restartPolicy: Never
```

   After you run the pod, the following output is returned:

```
SPECIAL_TYPE
SPECIAL_LEVEL
```

7.3.3. Manage Secrets

This topic describes how to manage Secrets in the Container Service for Kubernetes (ACK) console.

Prerequisites

An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.

Context

We recommend that you use Secrets to store sensitive information in Kubernetes clusters. The information includes passwords and certificates.

Secrets are classified into the following types:

- Service account: A service account is automatically created by Kubernetes and automatically mounted to the */run/secrets/kubernetes.io/serviceaccount* directory of a pod. The service account provides an identity for the pod to interact with the API server.
- Opaque: This type of Secret is encoded in Base64 and used to store sensitive information, such as passwords and certificates.

By default, you can create only Opaque Secrets in the ACK console. Opaque Secrets store map type data. Therefore, values must be encoded in Base64. You can create Secrets in the ACK console with a few clicks. Plaintext is automatically encoded in Base64.

You can also create Secrets by using the CLI. For more information, see Kubernetes Secrets.

Create a Secret

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > Secrets**.
5. On the **Secrets** page, click **Create** in the upper-right corner.
6. In the **Create** panel, configure the Secret.

   Configure the parameters described in the following table.

Parameter	Description
Name	Enter a name for the Secret. The name must be 1 to 253 characters in length, and can contain only lowercase letters, digits, hyphens (-), and periods (.).
Namespace	Specify the namespace of the Secret.
Type	You can select Opaque, Private Repository Logon Secret, and TLS Certificate.
Opaque	If you set Type to Opaque, configure the following parameters: ○ (Optional)To enter Secret data in plaintext, select **Encode Data Values Using Base64**. ○ Configure the Secret in key-value pairs. Click **+ Add**. Enter the keys and values for the Secret in the **Name** and **Value** fields.
Private Repository Logon Secret	If you set Type to Private Repository Logon Secret, configure the following parameters: ○ Docker Registry URL: Enter the address of the Docker registry where the Secret is stored. ○ Username: Enter the username that is used to log on to the Docker registry. ○ Password: Enter the password that is used to log on to the Docker registry.
TLS Certificate	If you set Type to TLS Certificate, configure the following parameters: ○ Cert: Enter a TLS certificate. ○ Key: Enter the key of the TLS certificate.

View a Secret

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > Secrets**.

5. On the **Secrets** page, find the Secret that you want to view and click **Details** in the **Actions** column.

   On the details page of the Secret, you can view details about the Secret. You can also view the key-value pairs in the Secret.

   > **Note** To view the values in plaintext, click the icon in the **Value** column.

Modify a Secret

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > Secrets**.
5. On the **Secrets** page, find the Secret that you want to modify and click **Edit** in the **Actions** column.
6. In the Edit panel, modify the Secret based on your requirements and click **OK**.

Delete a Secret

> **Note** Do not delete Secrets that are created together with the cluster.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Configurations > Secrets**.
5. On the **Secrets** page, find the Secret that you want to delete and click **Delete** in the **Actions** column. In the **Confirm** message that appears, click **OK**.

7.3.4. Use a Secret in a pod

We recommend that you use Secrets to store sensitive information in Kubernetes clusters. The information includes passwords and certificates. This topic describes how to create a Secret in the Container Service for Kubernetes (ACK) console. This topic also describes how to mount a Secret as a volume to a pod and expose a Secret as an environment variable for a pod. You can perform the operations by using the console or a CLI.

Prerequisites

- The Secret and pod are in the same cluster and belong to the same namespace.
- You are connected to a master node of the cluster. For more information, see Connect to Kubernetes clusters by using kubectl.

Context

You can use a Secret in a pod in the following scenarios:

- Mount a Secret as a volume to a pod.
- Expose a Secret as an environment variable for a pod.

For more information about Secrets, see Secrets.

Create a Secret

The following example shows how to create a Secret named secret-test.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. In the upper-right corner of the **Deployments** page, click **Create from YAML**.
6. Select a cluster and a namespace, select a sample template or enter a custom template, and then click **Create**.

   The following YAML template provides an example on how to create the Secret named secret-test:

```
apiVersion: v1
kind: Secret
metadata:
 name: secret-test
type: Opaque
data:
 username: admin
 password: 12345  #The value must be encoded in Base64.
```

For more information about how to create a Secret in the ACK console, see Manage Secrets.

Mount a Secret as a volume to a pod

You can mount a Secret as a volume to a pod by using the following methods:

Mount a Secret as a volume to a pod by using a CLI

A mounted Secret can be used as a file in a pod. In this example, the secret-test Secret that contains the username and password information is stored as a file under the */srt* directory.

1. Create an *example0.yaml* file and copy the following content into the file:

```
apiVersion: v1
kind: Pod
metadata:
 name: pod0
spec:
 containers:
 - name: redis
   image: redis
   volumeMounts:
   - name: srt
     mountPath: "/srt "
     readOnly: true
 volumes:
 - name: srt
   secret:
     secretName: secret-test
```

2. Run the following command to create a pod to which the secret-test Secret is mounted:

```
kubectl apply -f example0.yaml
```

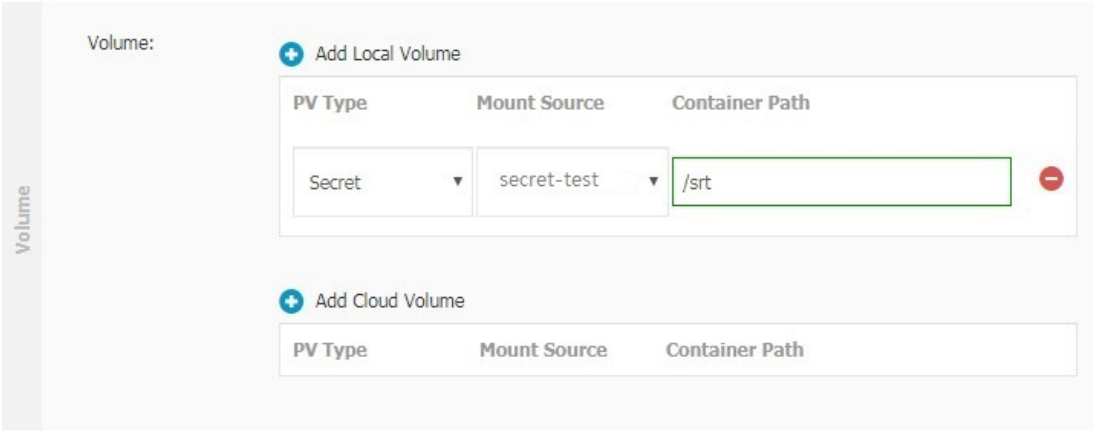
> **Note** Replace *example0.yaml* with the name of the YAML file that is used.

Mount a Secret as a volume to a pod in the ACK console

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. In the upper-right corner of the **Deployments** page, click **Create from Image**.

   For more information, see Create a stateless application by using a Deployment.
6. On the **Basic Information** wizard page, set the parameters and click **Next**.
7. On the **Container** wizard page, click **Add Local Storage** in the **Volume** section. Select *Secret* from the PV Type drop-down list, select the Secret that is created in Create a Secret from the Mount Source drop-down list, and specify a container path in the Container Path column. Click **Next**.

   The following figure shows an example on how to configure the volume.



8. On the **Advanced** wizard page, set the parameters and click **Create**.

Expose a Secret as an environment variable for a pod

You can expose a Secret as an environment variable for a pod by using the following methods:

Expose a Secret as an environment variable for a pod by using a CLI

In this example, the username and password stored in the secret-test Secret are referenced in an environment variable of a pod.

1. Create an *example1.yaml* file and copy the following content into the file:

```
apiVersion: v1
kind: Pod
metadata:
 name: pod1
spec:
 containers:
 - name: redis
  image: redis
  env:
   - name: USERNAME
    valueFrom:
     secretKeyRef:
      name: secret-test
      key: username
   - name: PASSWORD
    valueFrom:
     secretKeyRef:
      name: secret-test
      key: password
```

2. Run the following command to configure an environment variable:

```
kubectl apply -f example1.yaml
```

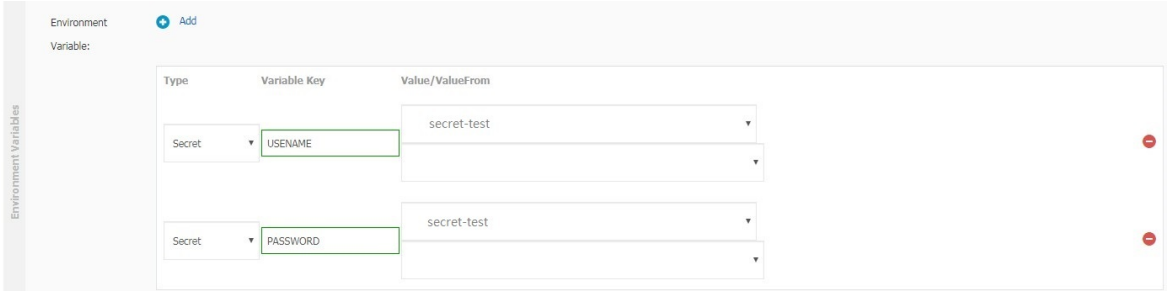
> **Note** Replace *example1.yaml* with the name of the YAML file that is used.

Expose a Secret as an environment variable for a pod in the ACK console

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. In the upper-right corner of the **Deployments** page, click **Create from Image**.

   For more information, see Create a stateless application by using a Deployment.
6. On the **Basic Information** wizard page, set the parameters and click **Next**.
7. On the **Container** wizard page, click  in the **Environments** section. In this example, select **Secret** from the Type drop-down list and select the Secret that is created in Create a Secret from the Value/ValueFrom drop-down list. After you select the Secret, you must specify the key of the key-value pair that you want to reference and specify a name for the environment variable.

   The following figure shows an example on how to configure the environment variable.



8. On the **Advanced** wizard page, set the parameters and click **Create**.

7.4. Schedule and deploy an application

7.4.1. Schedule pods to specific nodes

In some cases, you may need to deploy pods on a specified node to meet your business requirements. You may also need to deploy pods on nodes that use standard SSDs. This topic describes how to schedule a pod to a specific node in the Container Service for Kubernetes (ACK) console. You can configure node labels or pod templates to schedule pods to specific nodes.

Prerequisites

创建Kubernetes托管版集群.

Context

You can configure node labels and set `nodeSelector` to schedule a pod to a specific node. For more information about how to set nodeSelector, see nodeSelector.

Step 1: Configure node labels

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the Nodes page, click **Manage Labels and Taints** in the upper-right corner to go to the **Manage Labels and Taints** page.
6. On the **Labels** tab, select the nodes that you want to manage and click **Add Label**.
7. In the **Add** dialog box, specify **Name** and **Value**, and then click **OK**.
   - **Name**: the name of the label. The name can contain letters, digits, hyphens (-), underscores (_), and periods (.).The name must start and end with a letter or a digit.
   - **Value**: the value of the label. The value can contain letters, digits, hyphens (-), underscores (_), and periods (.).The label value must start and end with a letter or a digit. You can leave this parameter empty.

Step 2: Schedule a pod to a specific node

1. In the left-side navigation pane of the ACK console, click **Clusters**.
2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
4. On the **Deployments** page, click **Create from Template** in the upper-right corner.
5. Configure the template and use the template to deploy a pod.
   - **Namespace**: Select the namespace to which the pod belongs. In this example, the **default** namespace is selected.
   - **Sample Template**: In this example, Custom is selected.

   The following YAML template is used to deploy the pod:

```
apiVersion: v1
kind: Pod
metadata:
 labels:
  name: hello-pod
 name: hello-pod
spec:
 containers:
  - image: nginx
    imagePullPolicy: IfNotPresent
    name: hello-pod
    ports:
     - containerPort: 8080
       protocol: TCP
    resources: {}
    securityContext:
     capabilities: {}
     privileged: false
    terminationMessagePath: /dev/termination-log
 dnsPolicy: ClusterFirst
 restartPolicy: Always
 nodeSelector:
  group: worker                ##This value must be the same as the node label that is added in Step 1.
 status: {}
```

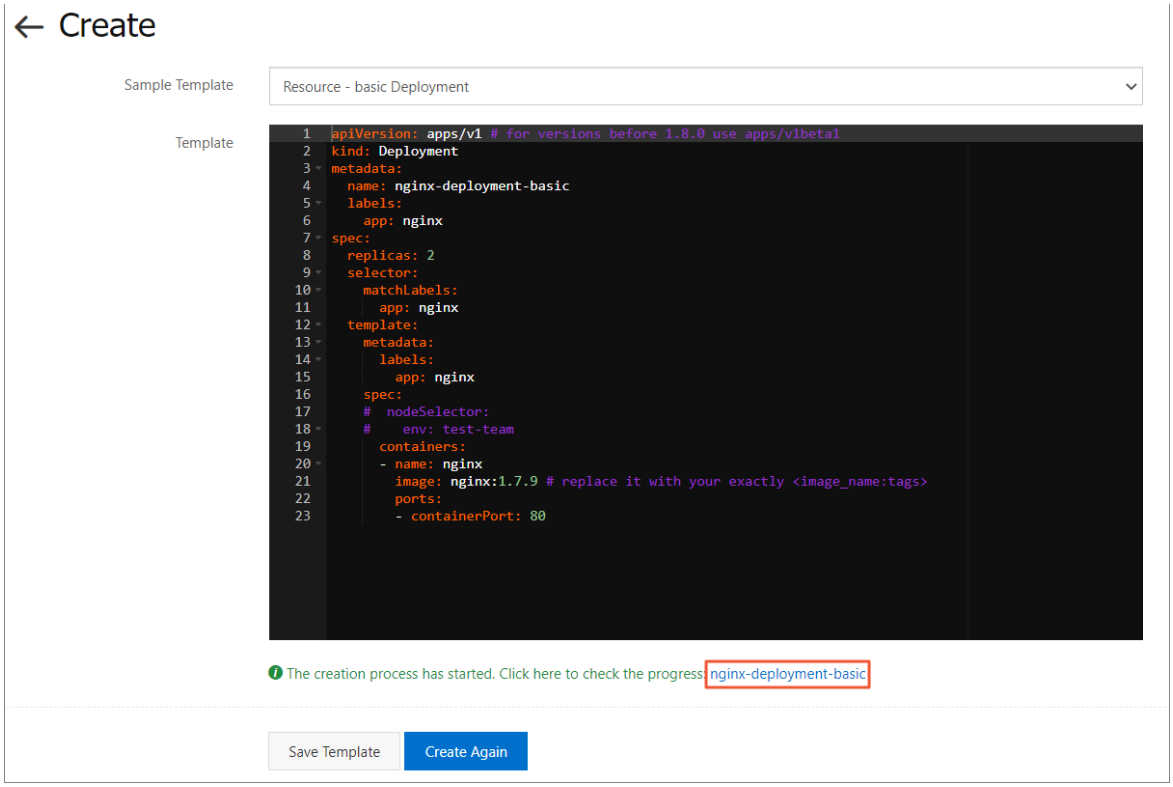
6. Click **Create**. A message that indicates the deployment status appears.

Verify the result

You can use one of the following methods to check whether the pod is deployed on the specified node:

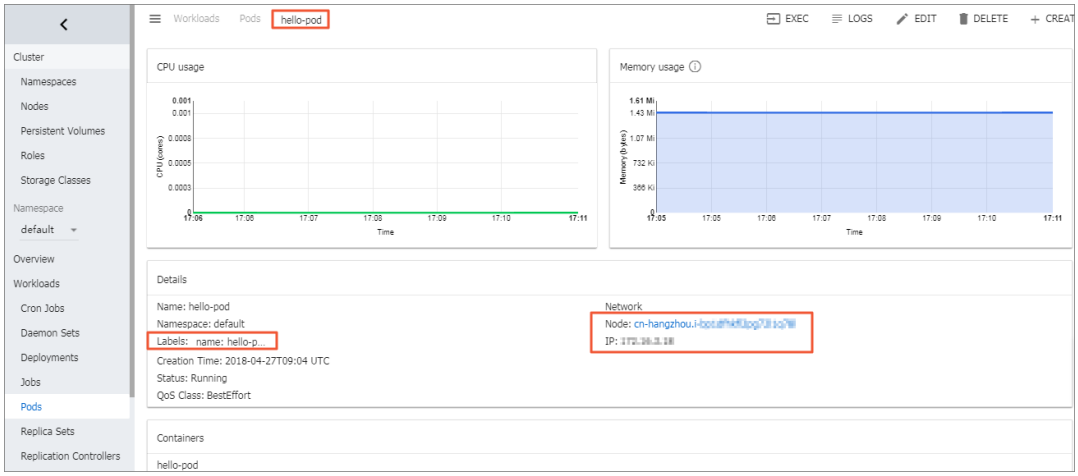
Method 1: Click the pod name after the pod is deployed

1. After the pod is deployed, click the pod name in the notification at the bottom of the page to go to the **Pods** page.



2. On the **Pods** page, click the pod name to go to the details page of the pod.

   You can view the labels of the pod and the ID of the node to which the pod is scheduled. The following figure indicates that the pod is scheduled to the node with the `group:worker` label.



Method 2: Check the containers on the Pods page

1. In the left-side navigation pane of the ACK console, click **Clusters**.
2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. In the left-side navigation pane of the details page, choose **Workloads > Pods**.
4. On the **Pods** page, click the pod name to go to the details page of the pod.

   You can view the labels of the pod and the ID of the node to which the pod is scheduled. The following figure indicates that the pod is scheduled to the node with the `group:worker` label.

7.4.2. Optimize pod scheduling by using descheduler

The descheduler component is used to schedule pods that cannot find suitable nodes. This optimizes pod scheduling, avoids resource waste, and improves resource utilization in Container Service for Kubernetes (ACK) clusters. This topic describes how to configure and use descheduler.

Prerequisites

- An ACK cluster of Kubernetes 1.14 or later is created. For more information, see 创建Kubernetes托管版集群.
- Helm 3 is installed. For more information, see Install Helm.
- kubectl is installed.

Procedure

1. Install descheduler by using Helm.
   i. Run the following command to add the chart repository of descheduler:

   ```
   helm repo add descheduler https://kubernetes-sigs.github.io/descheduler/
   ```

   Expected output:

   ```
   "descheduler" has been added to your repositories
   ```

   ii. Run the following command to install descheduler:

   ```
   helm install descheduler --namespace kube-system descheduler/descheduler
   ```

   Expected output:

   ```
   NAME: descheduler
   LAST DEPLOYED: Thu Mar  4 19:29:23 2021
   NAMESPACE: kube-system
   STATUS: deployed
   REVISION: 1
   TEST SUITE: None
   NOTES:
   Descheduler installed as a cron job.
   ```

   > **Note** You may experience timeout errors when you pull the default image of descheduler due to network issues. In this case, we recommend that you replace the default image with the image provided by Alibaba Cloud: `registry.cn-hangzhou.aliyuncs.com/carsnow/descheduler:v0.20.0`.

2. Check the installation progress.

   After the descheduler chart is installed, a CronJob is automatically created in the `kube-system` namespace. By default, this CronJob is configured to run every 2 minutes.

   ```
   kubectl get cronjob -n kube-system
   ```

   Expected output:

   ```
   NAME     SCHEDULE    SUSPEND  ACTIVE  LAST SCHEDULE  AGE
   descheduler  */2 * * * *  False   0    21s       47h
   ```

3. Check the scheduling policy of descheduler.

```
kubectl describe cm descheduler -n kube-system
```

Expected output:

```
Name:         descheduler
Namespace:    kube-system
Labels:       app.kubernetes.io/instance=descheduler
              app.kubernetes.io/managed-by=Helm
              app.kubernetes.io/name=descheduler
              app.kubernetes.io/version=0.20.0
              helm.sh/chart=descheduler-0.20.0
Annotations:  meta.helm.sh/release-name: descheduler
              meta.helm.sh/release-namespace: kube-system
Data
====
policy.yaml:
----
apiVersion: "descheduler/v1alpha1"
kind: "DeschedulerPolicy"
strategies:
 "RemoveDuplicates":
   enabled: true
 "RemovePodsViolatingInterPodAntiAffinity":
   enabled: true
 "LowNodeUtilization":
   enabled: true
   params:
    nodeResourceUtilizationThresholds:
     thresholds:
       "cpu" : 20
       "memory": 20
       "pods": 20
     targetThresholds:
       "cpu" : 50
       "memory": 50
       "pods": 50
 "RemovePodsHavingTooManyRestarts":
   enabled: true
   params:
    podsHavingTooManyRestarts:
     podRestartThreshold: 100
     includingInitContainers: true
Events:  <none>
```

For more information about the policy settings in the `strategies` section, see Descheduler.

4. Verify pod scheduling before the scheduling policy is modified.

i. Create a Deployment to test the scheduling.

Create a file named nginx.yaml and copy the following content into the file:

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
 name: nginx-deployment-basic
 labels:
  app: nginx
spec:
 replicas: 3
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
    ports:
    - containerPort: 80
```

Run the following command to deploy a Deployment with the *nginx.yaml* file:

```
kubectl apply -f nginx.yaml
```

Expected output:

```
deployment.apps/nginx-deployment-basic created
```

ii. Wait 2 minutes and run the following command to check the nodes to which the pods are scheduled:

```
kubectl get pod -o wide | grep nginx
```

Expected output:

```
NAME                          READY  STATUS   RESTARTS  AGE  IP          NODE                NOMINATED NODE  READI
NESS GATES
nginx-deployment-basic-**1  1/1   Running  0     36s  172.25.XXX.XX1  cn-hangzhou.172.16.XXX.XX2  <
none>      <none>
nginx-deployment-basic-**2  1/1   Running  0     11s  172.25.XXX.XX2  cn-hangzhou.172.16.XXX.XX3  <
none>      <none>
nginx-deployment-basic-**3  1/1   Running  0     36s  172.25.XXX.XX3  cn-hangzhou.172.16.XXX.XX3  <
none>      <none>
```

The output shows that pod `nginx-deployment-basic-**2` and pod `nginx-deployment-basic-**3` are scheduled to the same node `cn-hangzhou.172.16.XXX.XX3`.

5. Modify the scheduling policy.

To avoid pod scheduling being affected by multiple scheduling strategies, modify the ConfigMap in Step 3 to retain only the RemoveDuplicates strategy.

> **Note** The RemoveDuplicates strategy ensures that pods managed by replication controllers are scheduled to different nods.

6. Verify pod scheduling after the scheduling policy is modified.
   i. Deploy a new scheduling policy.

      Create a file named *newPolicy.yaml* and copy the following content into the file:

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: descheduler
 namespace: kube-system
 labels:
  app.kubernetes.io/instance: descheduler
  app.kubernetes.io/managed-by: Helm
  app.kubernetes.io/name: descheduler
  app.kubernetes.io/version: 0.20.0
  helm.sh/chart: descheduler-0.20.0
 annotations:
  meta.helm.sh/release-name: descheduler
  meta.helm.sh/release-namespace: kube-system
data:
 policy.yaml: |-
  apiVersion: "descheduler/v1alpha1"
  kind: "DeschedulerPolicy"
  strategies:
   "RemoveDuplicates":
     enabled: true
```

      Run the following command to apply the configurations in *newPolicy.yaml*:

```
kubectl apply -f newPolicy.yaml
```

      Expected output:

```
deployment.apps/nginx-deployment-basic created
```

   ii. Wait 2 minutes and run the following command to check the nodes to which the pods are scheduled:

```
kubectl get pod -o wide | grep nginx
```

      Expected output:

```
NAME                          READY  STATUS   RESTARTS  AGE   IP          NODE            NOMINATED NODE  REA
DINESS GATES
nginx-deployment-basic-**1  1/1   Running  0     8m26s  172.25.XXX.XX1  cn-hangzhou.172.16.XXX.XX2
<none>      <none>
nginx-deployment-basic-**2  1/1   Running  0     8m1s   172.25.XXX.XX2  cn-hangzhou.172.16.XXX.XX1
<none>      <none>
nginx-deployment-basic-**3  1/1   Running  0     8m26s  172.25.XXX.XX3  cn-hangzhou.172.16.XXX.XX3
<none>      <none>
```

      The output shows that pod `nginx-deployment-basic-**2` is rescheduled to `cn-hangzhou.172.16.XXX.XX1` by descheduler. In this case, each of the three test pods is scheduled to a different node. This balances pod scheduling among multiple nodes.

7.4.3. Use an application trigger to redeploy an application

Container Service Kubernetes (ACK) supports application triggers. You can use application triggers for different methods. This topic describes how to use an application trigger to redeploy an application.

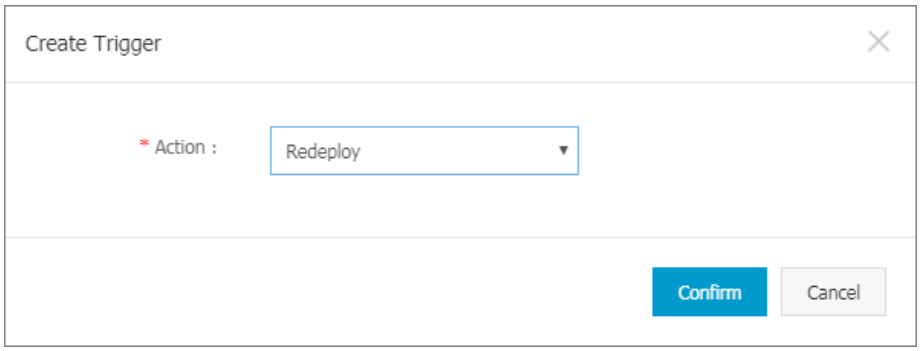
Prerequisites

- An ACK cluster is created. For more information, see Create a dedicated Kubernetes cluster.
- An application is created. You can use this application to create and test a trigger. In this example, an NGINX application is created.

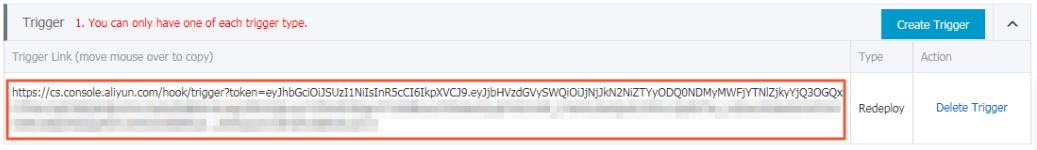
Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the Deployments tab, set the namespace, find the application that you want to manage, and then click **Details** in the **Actions** column of the application.
6. On the NGINX application details page, click the **Triggers** tab and click **Create Trigger**.
7. In the **Create Trigger** dialog box, set **Action** to **Redeploy** and click **OK**.

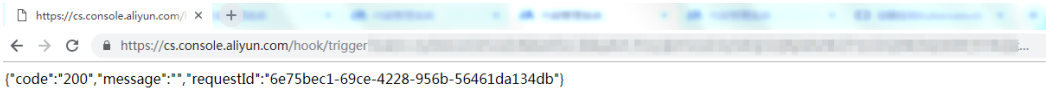
   Note Only Redeploy is available in the **Action** drop-down list.



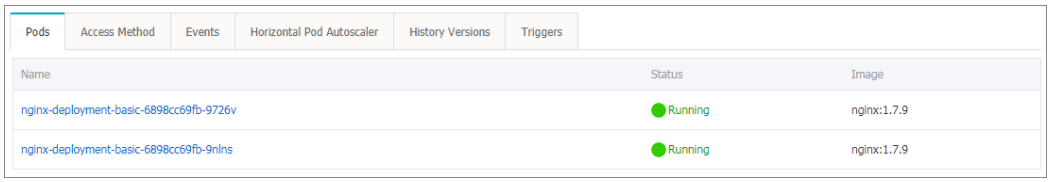
   After the trigger is created, a trigger link appears in the Trigger Link Address column on the Triggers tab.



8. Copy the link and open it in your browser. A message appears to display specific information such as the request ID.



9. Go to the NGINX application details page. A new pod appears on the page.



After the new pod is deployed, the original pod is automatically deleted.

What's next

You can call a trigger by using GET or POST in a third-party system. For example, you can run the **curl** command to call a trigger.

To call the trigger to redeploy an application, run the following command:

```
curl https://cs.console.aliyun.com/hook/trigger?token=xxxxxxx
```

7.4.4. Use Helm to simplify application deployment

This topic introduces the basic concepts of Helm and describes how to use Helm to deploy an Apache Spark-based WordPress application in a Container Service for Kubernetes (ACK) cluster.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
  Tiller is automatically deployed to the cluster when the ACK cluster is created. Helm command-line interface (CLI) is automatically installed on each master node. An Alibaba Cloud chart repository is added to Helm.
- A Kubernetes version that supports Helm is used.
  Only Kubernetes 1.8.4 and later support Helm. If the Kubernetes version of your cluster is V1.8.1, you can **upgrade** the cluster on the Clusters page of the ACK console.

Context

Application management is the most challenging task in Kubernetes. The Helm project provides a unified method to package software and manage software versions. You can use Helm to simplify application distribution and deployment. App Catalog is integrated with Helm in the ACK console and provides extended features based on Helm. App Catalog also supports Alibaba Cloud chart repositories to help you accelerate application deployments. You can deploy applications in the ACK console or by using Helm CLI.

Basic concepts of Helm

Helm is an open source project initiated by Deis. Helm can be used to simplify the deployment and management of Kubernetes applications.

Helm serves as a package manager for Kubernetes and allows you to find, share, and use applications built by using Kubernetes. When you use Helm, you must review the following concepts:

- Chart: a packaging format used by Helm. Each chart contains the images, dependencies, and resource definitions that are required to run an application. A chart may contain service definitions in a Kubernetes cluster. A Helm chart is similar to a Homebrew formula, an Advanced Package Tool (APT) dpkg, or a Yum rpm.
- Release: an instance of a chart that runs in a Kubernetes cluster. A chart can be installed multiple times into a Kubernetes cluster. After a chart is installed, a new release is created. For example, you can install a MySQL chart. If you want to run two databases in your cluster, you can install the MySQL chart twice. Each time a chart is installed, a release is created with a different name.

- Repository: the storage of charts. Charts are published and stored in repositories.

Helm components

Helm uses a client-server architecture and consists of the following components:

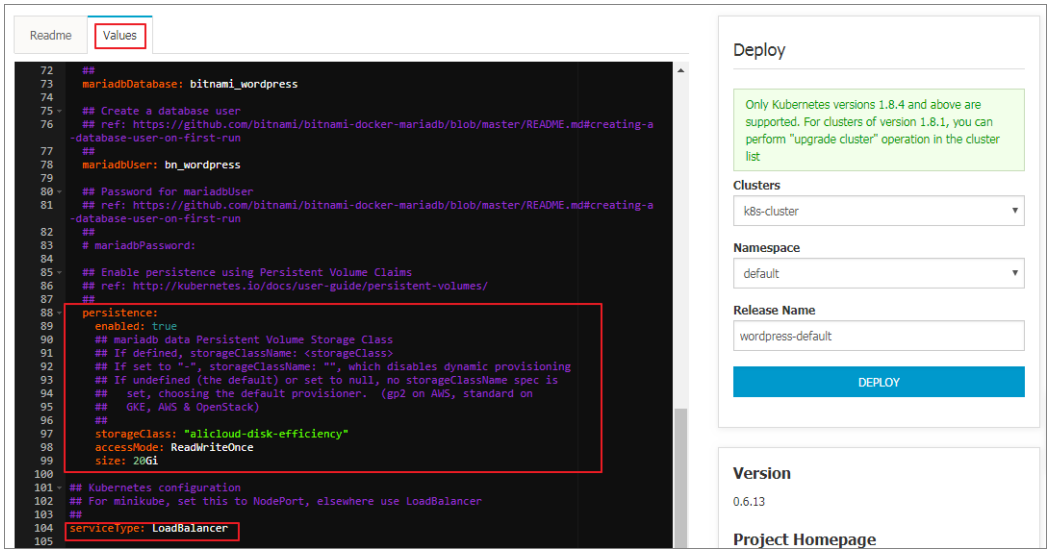
- Helm CLI is the Helm client that runs on your on-premises machine or on the master node of a Kubernetes cluster.
- Tiller is the server-side component and runs in a Kubernetes cluster. Tiller manages the lifecycles of Kubernetes applications.
- A repository is used to store charts. The Helm client can access the index file and packaged charts in a chart repository over HTTP.

Deploy an application in the ACK console

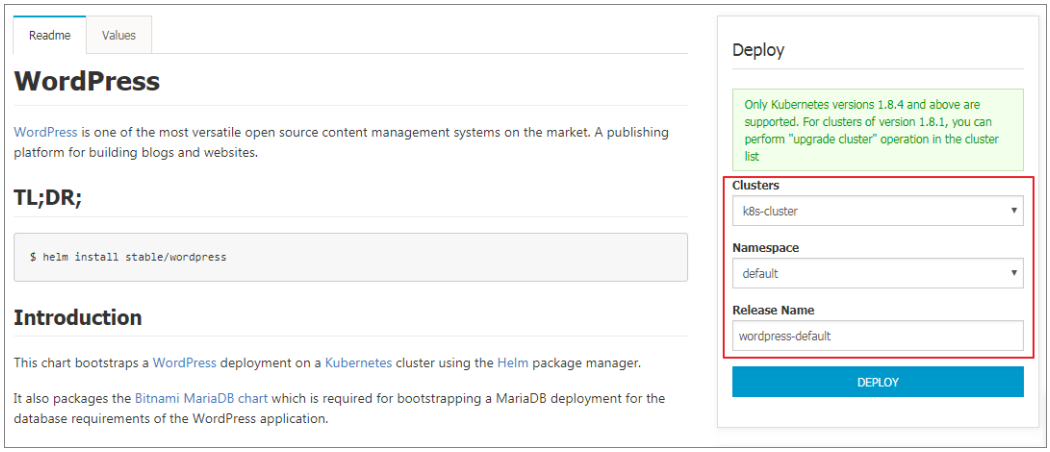
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **Alibaba Cloud Apps** tab, select and click a chart to go to the details page. In this example, **ack-wordpress-sample** is used.
4. Click the **Parameters** tab and modify the parameters.

   In this example, a persistent volume claim (PVC) is specified to bind a dynamic disk volume. For more information, see Usage notes for disk volumes.

   > **Note** You must first provision a disk as a persistent volume (PV). The volume size claimed in the PVC cannot exceed the capacity of the PV.



5. On the right side of the page, configure the basic settings and click **Create**.
   - **Cluster**: the cluster where you want to deploy the application.
   - **Namespace**: the namespace where you want to deploy the application. By default, the default namespace is selected.
   - **Release Name**: the release name for the application.



6. In the left-side navigation pane of the details page, choose **Network > Services**.
7. In the Service list, you can find the Service created for the application and the corresponding external endpoints through HTTP and HTTPS. Click an external endpoint to access the WordPress blog page.

> **Note** Before you access the external endpoint, make sure that the port of the endpoint is added to the security group.

Deploy an application by using Helm CLI

After Helm CLI is automatically installed in the ACK cluster and the required chart repository is added to Helm, you can log on to the cluster by using SSH. Then, you can deploy applications by using Helm CLI. For more information, see Use SSH to connect to an ACK cluster. You can also install and configure Helm CLI and kubectl on your on-premises machine.

In this example, Helm CLI and kubectl are installed and configured on your on-premises machine, and then an Apache Spark-based WordPress application is deployed.

1. Install and configure Helm CLI and kubectl.
   i. Install and configure kubectl on your on-premises machine.

      For more information, see Connect to Kubernetes clusters by using kubectl.
      To view the details of a Kubernetes cluster, run the `kubectl cluster-info` command.

   ii. Install Helm on your on-premises machine.

      For more information, see Install Helm.
2. Deploy a WordPress application.

   In the following example, a WordPress blog website is deployed by using Helm.

   i. Run the following command:

   ```
   helm install --name wordpress-test stable/wordpress
   ```

   > **Note** ACK supports dynamic disk volumes. You must first provision a disk as a PV.

   The following output is returned:

   ```
   NAME:  wordpress-test
   LAST DEPLOYED: Mon Nov  20 19:01:55 2017
   NAMESPACE: default
   STATUS: DEPLOYED
   ...
   ```

ii. Run the following commands to query the release and Service created for the WordPress application:

```
helm list
kubectl get svc
```

iii. Run the following command to view the pods provisioned for the WordPress application. You may need to wait before the pods change to the Running state.

```
kubectl get pod
```

iv. Run the following command to obtain the endpoint of the WordPress application:

```
echo http://$(kubectl get svc wordpress-test-wordpress -o jsonpath='{.status.loadBalancer.ingress[0].ip }')
```

You can enter the preceding URL into the address bar of your browser to access the WordPress application.
You can also run the following commands based on the chart description to obtain the username and password of the administrator for the WordPress application:

```
echo Username: user
echo Password: $(kubectl get secret --namespace default wordpress-test-wordpress -o jsonpath="{.data.wordpress-password}" | base64 --decode)
```

v. To delete the WordPress application, run the following command:

```
helm delete --purge wordpress-test
```

Use a third-party chart repository

You can use the default Alibaba Cloud chart repository. If a third-party chart repository is accessible from your cluster, you can also use the third-party chart repository. Run the following command to add a third-party chart repository to Helm:

```
helm repo add Repository name Repository URL
helm repo update
```

For more information about Helm commands, see Helm documentation.

References

Helm contributes to the development of Kubernetes. A growing number of software suppliers, such as Bitnami, have provided high-quality charts. For more information about available charts, visit `https://kubeapps.com/` .

7.4.5. Use OpenKruise to deploy cloud-native applications

OpenKruise is a standard extension of Kubernetes and can work with Kubernetes-native clusters. Automation is the core feature of OpenKruise that allows you to automate the deployment of applications in Kubernetes, including pod deployment, upgrades, and scaling. This topic describes how to use OpenKruise to deploy cloud-native applications.

Context

OpenKruise is an open source automation engine provided by Alibaba Cloud for cloud-native applications. It is used as a deployment base to migrate the business of Alibaba Group to the cloud. OpenKruise has joined the Cloud Native Computing Foundation (CNCF) Sandbox project.

OpenKruise contains a variety of custom workloads. You can use the workloads to deploy and manage stateless applications, stateful applications, sidecar containers, and daemon applications. OpenKruise also supports advanced strategies such as in-place upgrades, canary release, stream upgrades, and priority configuration.

Instruction

OpenKruise provides controllers such as CloneSet, Advanced StatefulSet, and Advanced DaemonSet. The following section describes the features of commonly used controllers.

Commonly used controllers

Controller	Feature	Start rating
CloneSet	CloneSets are equivalent to Kubernetes-native Deployments. CloneSets are used to manage stateless applications. For more information, see CloneSet. The fields in a CloneSet YAML file do not completely match those in a Deployment YAML file. However, CloneSets support all features of Deployments and provide more strategies.	☆☆☆☆☆
Advanced StatefulSet	Advanced StatefulSets are equivalent to Kubernetes-native StatefulSets. Advanced StatefulSets are used to manage stateful applications. For more information, see Advanced StatefulSet. The fields in an Advanced StatefulSet YAML file completely match those in a StatefulSet YAML file. You only need to change the value of `apiVersion` to `apps.kruise.io/v1alpha1`. In addition, you can set the `optional` field to use more release strategies, such as in-place upgrades and parallel release.	☆☆☆☆
Advanced DaemonSet	Advanced DaemonSets are equivalent to Kubernetes-native DaemonSets. Advanced DaemonSets are used to manage daemon applications. For more information, see Advanced DaemonSet. The fields in an Advanced DaemonSet YAML file completely match those in a DaemonSet YAML file. You only need to change the value of `apiVersion` to `apps.kruise.io/v1alpha1`. In addition, you can set the `optional` field to use more release strategies, such as hot upgrades, canary release, and canary release by node label.	☆☆☆☆

Controller	Feature	Start rating
SidecarSet	The SidecarSet controller independently manages sidecar containers and injects sidecar containers to pods. For more information, see SidecarSet. After you define a sidecar container and a label selector in an independent custom resource (CR), OpenKruise injects the defined sidecar container to the pod that matches the conditions when the pod is created. You can also perform in-place upgrades for the injected sidecar container by using a SidecarSet.	☆☆☆☆
UnitedDeployment	The UnitedDeployment controller manages multiple sub-workloads in different regions. For more information, see UnitedDeployment. The UnitedDeployment controller supports the following sub-workloads: CloneSets, StatefulSets, and Advanced StatefulSets. You can use one UnitedDeployment to manage sub-workloads in different regions and pod replicas of these sub-workloads.	☆☆☆

The following section compares the CloneSet, Advanced StatefulSet, and Advanced DaemonSet controllers of OpenKruise with the corresponding controllers provided by the Kubernetes community.

Feature comparison

Feature	CloneSet VS Deployment		Advanced StatefulSet VS StatefulSet		Advanced DaemonSet VS DaemonSet	
	CloneSet	Deployment	Advanced StatefulSet	StatefulSet	Advanced DaemonSet	DaemonSet
Stream scaling	Not supported (coming soon)	Not supported	Not supported	Not supported	Supported	Not supported
Specified pod deletion	Supported	Not supported	Supported	Not supported	Not supported	Not supported
Upgrade pods upon recreation	Supported	Supported	Supported	Supported	Supported	Supported
In-place pod upgrades	Supported	Not supported	Supported	Not supported	Not supported (coming soon)	Not supported
Canary release	Supported	Not supported	Supported	Supported	Supported	Not supported

Feature	CloneSet VS Deployment		Advanced StatefulSet VS StatefulSet		Advanced DaemonSet VS DaemonSet	
	CloneSet	Deployment	Advanced StatefulSet	StatefulSet	Advanced DaemonSet	DaemonSet
MaxUnavailable	Supported	Supported	Supported	Not supported	Supported	Supported
MaxSurge	Supported	Supported	Not supported	Not supported	Supported	Not supported
Customizing release sequence by using the priority or scattering strategy	Supported	Not supported	Supported	Not supported	Supported	Not supported
Use the lifecycle hook to manage the lifecycle of pods	Supported	Not supported	Not supported	Not supported	Not supported	Not supported

Install OpenKruise

You can install OpenKruise by using Container Service for Kubernetes (ACK) App Catalog or Helm charts. We recommend that you use Helm charts to install OpenKruise. The following is a comparison of the two installation methods:

> **Notice** Before you install OpenKruise, make sure that the Kubernetes version is 1.13 or later. If you use Kubernetes 1.13 or 1.14, you must enable `CustomResourceWebhookConversion` feature-gate in kube-apiserver before you install OpenKruise.

- Install OpenKruise with ACK App Catalog: You can install OpenKruise with one click. You do not need to use the Helm command-line interface (CLI). However, after you install OpenKruise, you can upgrade it only by running the `helm upgrade` command.
- Install OpenKruise with Helm charts: This method is applicable to all Kubernetes-native clusters. If you choose this method, you can manage versions and configure parameters as needed. However, to install OpenKruise with Helm charts, you must use the CLI.

Install OpenKruise with ACK App Catalog

1.
2.
3. On the **Alibaba Cloud Apps** tab, click **Application Management** and click **ack-kruise**.
4. On the **App Catalog - ack-kruise** page, set the **Cluster**, **Namespace**, and **Release Name** parameters in the **Deploy** section, and then click **Create**.

Install OpenKruise with Helm charts

1. Install the CLI of Helm. For more information about how to download the CLI, see Helm Release.

> **Note** Make sure that the version of the Helm CLI that you install is 3.1.0 or later.

2. Install OpenKruise.

> **Note** If you want to configure OpenKruise, see Install OpenKruise.

   - If you use Kubernetes 1.13 or 1.14, run the following command to install OpenKruise:

```
Kubernetes 1.13 or 1.14
helm install kruise https://github.com/openkruise/kruise/releases/download/v0.7.0/kruise-chart.tgz --disable-openapi-validation
```

   - If you use Kubernetes 1.15 or later, run the following command to install OpenKruise:

```
Kubernetes 1.15 or later
helm install kruise https://github.com/openkruise/kruise/releases/download/v0.7.0/kruise-chart.tgz
```

Use a CloneSet to deploy a stateless application

1. Create a CloneSet.
   i. Create a *cloneset.yaml* file.

```
apiVersion: apps.kruise.io/v1alpha1
kind: CloneSet
metadata:
 name: demo-clone
spec:
 replicas: 5
 selector:
  matchLabels:
   app: guestbook
 template: #The schema of the pod template is the same as that of a Deployment.
  metadata:
   labels:
    app: guestbook
  spec:
   affinity:
    podAntiAffinity:
     preferredDuringSchedulingIgnoredDuringExecution:
     - podAffinityTerm:
        labelSelector:
         matchExpressions:
         - key: app
          operator: In
          values:
          - guestbook
        topologyKey: kubernetes.io/hostname
       weight: 100
   containers:
   - name: guestbook
    image: registry.cn-hangzhou.aliyuncs.com/kruise-test/guestbook:v1
    env:
    - name: test
     value: foo
 updateStrategy:
  type: InPlaceIfPossible    #We recommend that you perform an in-place upgrade instead of upgrading pods upon recreation.
  maxUnavailable: 20%      #A maximum of 20% pods can be unavailable during the release.
  inPlaceUpdateStrategy:
   gracePeriodSeconds: 3   #The graceful period specifies how long a pod stays in the Not-ready state before the controller upgrades the pod in place.
```

- type: specifies the upgrade strategy. The following three strategies are supported:
  - ReCreate: The controller deletes the current pods and persistent volume claims (PVCs), and then creates new pods and PVCs.
  - InPlaceIfPossible: The controller attempts to perform an in-place upgrade. If the attempt fails, the controller upgrades the pods by recreating them.
  - InPlaceOnly: The controller is allowed to perform only in-place upgrades.
- maxUnavailable: The maximum number of pods that can be unavailable during the upgrade process. You can specify an absolute value or a percentage value.
- gracePeriodSeconds: The graceful period specifies how long a pod stays in the Not-ready state before the controller upgrades the pod in place.

ii. Make the *cloneset.yaml* file effective in the ACK cluster.

```
kubectl create -f ./cloneset.yaml
```

Expected output:

```
cloneset.apps.kruise.io/demo-clone created
```

2. Query the states of the pods.

```
kubectl get pod
```

Expected output:

```
NAME         READY  STATUS   RESTARTS  AGE
demo-clone-5b9kl  1/1   Running  0      3s
demo-clone-6xjdg  1/1   Running  0      3s
demo-clone-bvmdj  1/1   Running  0      3s
demo-clone-dm22s  1/1   Running  0      3s
demo-clone-rbpg9  1/1   Running  0      3s
```

3. View the CloneSet.

```
kubectl get clone
```

Expected output:

```
NAME    DESIRED  UPDATED  UPDATED_READY  READY  TOTAL  AGE
demo-clone  5     5     5        5    5    46s
```

- DESIRED: the expected number of pods (spec.replicas).
- UPDATED: the number of pods that are upgraded (status.updatedReplicas).
- UPDATED_READY: the number of pods that are available after they are upgraded (status.updatedReadyReplicas).
- READY: the total number of available pods (status.readyReplicas).
- TOTAL: the total number of pods (status.replicas).

Use an Advanced StatefulSet to deploy a stateful application

1. Create an Advanced StatefulSet.

i. Create a *statefulset.yaml* file.

```
apiVersion: apps.kruise.io/v1alpha1
kind: StatefulSet
metadata:
 name: demo-asts
spec:
 replicas: 3
 selector:
  matchLabels:
   app: guestbook-sts
 podManagementPolicy: Parallel
 template: #The schema of the pod template is the same as that of a StatefulSet YAML.
  metadata:
   labels:
    app: guestbook-sts
  spec:
   affinity:
    podAntiAffinity:
     preferredDuringSchedulingIgnoredDuringExecution:
     - podAffinityTerm:
        labelSelector:
         matchExpressions:
         - key: app
           operator: In
           values:
           - guestbook-sts
        topologyKey: kubernetes.io/hostname
       weight: 100
   containers:
   - name: guestbook
    image: registry.cn-hangzhou.aliyuncs.com/kruise-test/guestbook:v1
    env:
    - name: test
     value: foo
    volumeMounts:
    - name: log-volume
     mountPath: /var/log
   readinessGates:
   - conditionType: InPlaceUpdateReady
   volumes:
   - name: log-volume
    emptyDir: {}
 updateStrategy:
  type: RollingUpdate
  rollingUpdate:
   podUpdatePolicy: InPlaceIfPossible #We recommend that you perform an in-place upgrade instead of upgrading pods upon recreation.
   maxUnavailable: 20% #A maximum of 20% pods can be unavailable during the release.
   inPlaceUpdateStrategy:
    gracePeriodSeconds: 3 #The graceful period specifies how long a pod stays in the Not-ready state before the controller upgrades the pod in place.
```

- type: specifies the upgrade strategy. The following three strategies are supported:
  - ReCreate: The controller deletes the current pods and PVCs, and then creates new pods and PVCs.
  - InPlaceIfPossible: The controller attempts to perform an in-place upgrade. If the attempt fails, the controller upgrades the pods by recreating them.
  - InPlaceOnly: The controller is allowed to perform only in-place upgrades.
- maxUnavailable: The maximum number of pods that can be unavailable during the upgrade process. You can specify an absolute value or a percentage value.
- gracePeriodSeconds: The graceful period specifies how long a pod stays in the Not-ready state before the controller upgrades the pod in place.

ii. Make the *statefulset.yaml* file effective in the ACK cluster.

```
kubectl create -f ./statefulset.yaml
```

Expected output:

```
statefulset.apps.kruise.io/demo-asts created
```

2. Query the states of the pods.

```
kubectl get pod
```

Expected output:

```
NAME     READY STATUS  RESTARTS AGE
demo-asts-0 1/1   Running 0     3h29m
demo-asts-1 1/1   Running 0     3h29m
demo-asts-2 1/1   Running 0     3h29m
```

3. View the Advanced StatefulSet.

```
kubectl get asts
```

Expected output:

```
NAME    DESIRED CURRENT UPDATED READY AGE
demo-asts 3     3    3    3   3h30m
```

- DESIRED: the expected number of pods (spec.replicas).
- UPDATED: the number of pods that are upgraded (status.updatedReplicas).
- READY: the total number of available pods (status.readyReplicas).
- TOTAL: the total number of pods (status.replicas).

Related information

- OpenKruise official documentation
- OpenKruise Github

7.4.6. Deploy, release, and monitor applications in the ACK console

The Container Service for Kubernetes (ACK) console provides extensive features to help you manage and maintain clusters and applications. This topic describes how to deploy an NGINX application, configure an Ingress, and query log data in the ACK console.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- Log Service is enabled and Logtail is installed. For more information, see Collect log files from containers by using Log Service.

Step 1: Deploy an NGINX application

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Configure the parameters and click **Create**.
   - In the upper-part of the page, set **Namespace** to **default**.
   - Set **Sample Template** to **Resource - basic Deployment**.
   - Configure log collection. For more information about how to configure log collection parameters, see Collect log files from containers by using Log Service.
     Add the following content to the `spec` field in the template.

     > **Note** `aliyun_logs_log-nginxvarlog` specifies that the name of the created Logstore is log-nginxvarlog. The value stdout specifies that the Logstore collects standard outputs of containers from the specified path.

     ```
     env:
     - name: aliyun_logs_log-nginxvarlog
       value: stdout
     ```

     ```
     16      spec:
     17      #  nodeSelector:
     18      #    env: test-team
     19        containers:
     20        - name: nginx
     21          image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
     22          ports:
     23          - containerPort: 80
     24          env:
     25          - name: aliyun_logs_log-nginxvarlog
     26            value: stdout
     ```

   - Change the version of the image that is used to deploy the NGINX application to 1.9.1.

     ```
     16      spec:
     17      #  nodeSelector:
     18      #    env: test-team
     19        containers:
     20        - name: nginx
     21          image: nginx:1.9.1 # replace it with your exactly <image_name:tags>
     22          ports:
     23          - containerPort: 80
     24          env:
     25          - name: aliyun_logs_log-nginxvarlog
     26            value: stdout
     ```

   After the NGINX application is created, you can view the application on the Deployments page.

Step 2: Release the NGINX application

1. In the left-side navigation pane of the details page, choose **Network > Services**.

2. On the **Services** page, click **Create** in the upper-right corner of the page.
3. In the **Create Service** dialog box, set the parameters and click **Create**.

Parameter	Description
Name	Enter a name for the Service.
Type	Set **Type** to **Cluster IP**.
Backend	Select **nginx-deployment-basic** from the drop-down list.
Port Mapping	Enter a name for the port. Set **Service Port** to 80, **Container Port** to 80, and **Protocol** to **TCP**.
Annotations	Add one or more annotations to the Service and configure Server Load Balancer (SLB) parameters. For example, the annotation `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20` specifies that the maximum bandwidth of the Service is 20 Mbit/s. This limits the amount of traffic that flows through the Service. For more information, see Use annotations to configure load balancing.
Label	Add one or more labels to the Service. Labels are used to identify the Service.

Step 3: Create an Ingress

1. In the left-side navigation pane of the details page, choose **Network > Ingresses**.
2. On the **Ingresses** page, click **Create** in the upper-right corner of the page.
3. In the **Create** dialog box, set the required parameters and click **Create**.

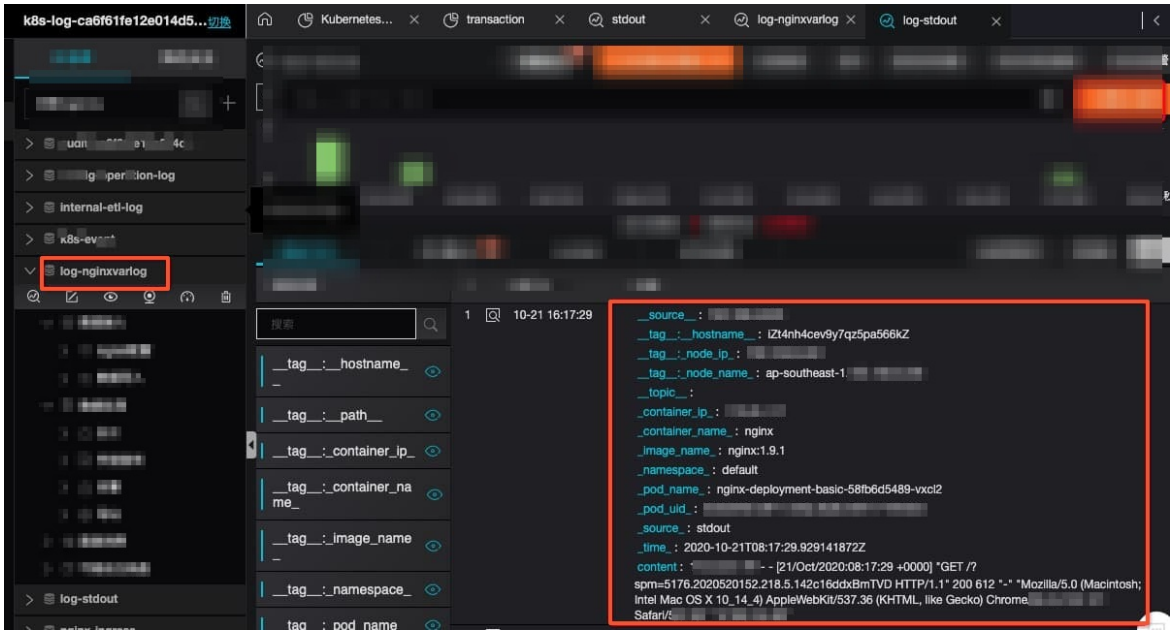
   The following section describes the key parameters. For more information about how to configure other parameters, see How to perform basic operations on an Ingress in the ACK console.

   - Name: Enter a name for the Ingress. The name is set to ingress-demo in this example.
   - Rules: Enter a custom domain name in the **Domain** field and enter the path */nginx* in the **Path** field. Set Name in the Services section to the name of the Service that you created in Step 2: Release the NGINX application and use the automatically matched port 80 for **Port**.

   On the **Ingresses** page, obtain the IP address of ingress-demo in the **Endpoint** column. Enter the *endpoint IP address* in the address bar of your browser. If the following page is displayed, it indicates that the Ingress is created.

Step 4: View the access log of the NGINX application

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, click **Cluster Information**.
5. On the page that appears, click the **Cluster Resources** tab, find **Log Service Project**, and then click its ID to check the Logstore.
6. On the Logstores tab, select log-nginxvarlog to view the latest access log of the application.



7.5. FAQ about application management

This topic provides answers to some frequently asked questions about application management.

- Troubleshoot application errors in Container Service for Kubernetes
- How do I manually upgrade Helm?
- How do I use private images in Kubernetes clusters?
- Precheck failure during a Cloud Controller Manager (CCM) upgrade
- How do I create containers from private images in an ACK cluster?
- Troubleshoot failures to bind source code in Container Registry
- Troubleshoot failures to create repositories in Container Registry

How do I manually upgrade Helm?

1. Log on to a master node in the Kubernetes cluster. For more information, see Connect to Kubernetes clusters by using kubectl.
2. Run the following command.

   For the image address, enter the domain name of the image in the region where the virtual private cloud (VPC) that hosts the image is deployed. For example, if your server is deployed in the China (Hangzhou) region, the image address is registry-vpc.cn-hangzhou.aliyuncs.com/acs/tiller:v2.11.0.

```
helm init --tiller-image registry.cn-hangzhou.aliyuncs.com/acs/tiller:v2.11.0 --upgrade
```

3. After the **tiller** health check succeeds, you can run the `helm version` command to view the upgrade results.

   > **Note**
   > - The preceding command only upgrades Tiller, the server-side component of Helm. To upgrade the client-side component, download the required client binary.
   > - Download the latest client version Helm client 2.11.0 that is supported by Alibaba Cloud.

4. After the client-side and server-side components of Helm are both upgraded, run the **helm version** command to view the following information:

```
helm version
```

```
Client: &version.Version{SemVer:"v2.11.0", GitCommit:"2e55dbe1fdb5fdb96b75ff144a339489417b146b", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.11.0", GitCommit:"2e55dbe1fdb5fdb96b75ff144a339489417b146b", GitTreeState:"clean"}
```

How do I use private images in Kubernetes clusters?

1. Run the following command:

```
kubectl create secret docker-registry regsecret --docker-server=registry-internal.cn-hangzhou.aliyuncs.com --docker-username=abc@aliyun.com --docker-password=xxxxxx --docker-email=abc@aliyun.com
```

   > **Note**
   > - `regsecret`: the name of the Secret. You can enter a custom name.
   > - `--docker-server`: the address of the Docker registry.
   > - `--docker-username`: the username of the Docker registry.
   > - `--docker-password`: the logon password of the Docker registry.
   > - `--docker-email`: the email address. This parameter is optional.

   You can use the private image by using one of the following two methods:

   - Manually configure the private image
     Add the Secret configuration to the YAML configuration file.

```
containers:
  - name: foo
    image: registry-internal.cn-hangzhou.aliyuncs.com/abc/test:1.0
imagePullSecrets:
  - name: regsecret
```

> **Note**
> - imagePullSecrets specifies the Secret that is required to pull the image.
> - regsecret must be the same as the previous configured Secret name.
> - The Docker registry address in image must be the same as the one that is specified in --docker-server.

For more information, see Use a private registry.

- Automatically configure the private image without the Secret

> **Note** To avoid referencing the Secret each time you use private images for deployment, you can add the Secret configuration to the default service account of the namespace. For more information, see Add ImagePullSecrets to a service account.

a. Run the following command to view the Secret that is required to pull the private image:

```
kubectl get secret regsecret
```

```
NAME     TYPE                DATA   AGE
regsecret  kubernetes.io/dockerconfigjson  1      13m
```

In this example, the default service account of the namespace is manually configured to use this Secret as the imagePullSecret.

b. Create an *sa.yaml* file and add the configuration of the default service account to this file.

```
kubectl get serviceaccounts default -o yaml > ./sa.yaml
cat sa.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
 creationTimestamp: 2015-08-07T22:02:39Z
 name: default
 namespace: default
 resourceVersion: "243024"         ## Take note of the selfLink field: /api/v1/namespaces/default/serviceaccounts/default
 uid: 052fb0f4-3d50-11e5-b066-42010af0d7b6
secrets:
- name: default-token-uudgeoken-uudge
```

c. In the command-line interface (CLI), enter `vim sa.yaml` to open the sa.yaml file, delete the resourceVersion parameter, and then add the imagePullSecrets parameter to specify the Secret for pulling images. The following sample code shows the modification:

```
apiVersion: v1
kind: ServiceAccount
metadata:
 creationTimestamp: 2015-08-07T22:02:39Z
 name: default
 namespace: default
 selfLink: /api/v1/namespaces/default/serviceaccounts/default
 uid: 052fb0f4-3d50-11e5-b066-42010af0d7b6
secrets:
- name: default-token-uudge
imagePullSecrets:          ## This field is newly added.
- name: regsecret
```

d. Use the configuration in the *sa.yaml* file to replace the configuration of the default service account.

```
kubectl replace serviceaccount default -f ./sa.yaml
serviceaccount "default" replaced
```

e. In the CLI, enter **kubectl create -f** to create a Tomcat application.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: tomcat-deployment
 labels:
  app: tomcat
spec:
 replicas: 1
 selector:
  matchLabels:
   app: tomcat
 template:
  metadata:
   labels:
    app: tomcat
  spec:
   containers:
   - name: tomcat
    image: registry-internal.cn-hangzhou.aliyuncs.com/abc/test:1.0          # Replace the value with the address of your private image
    - containerPort: 8080
```

f. If the configuration is correct, the pod is started. In the CLI, enter **kubectl get pod tomcat-xxx -o yaml**. You can find the following configuration in the command output:

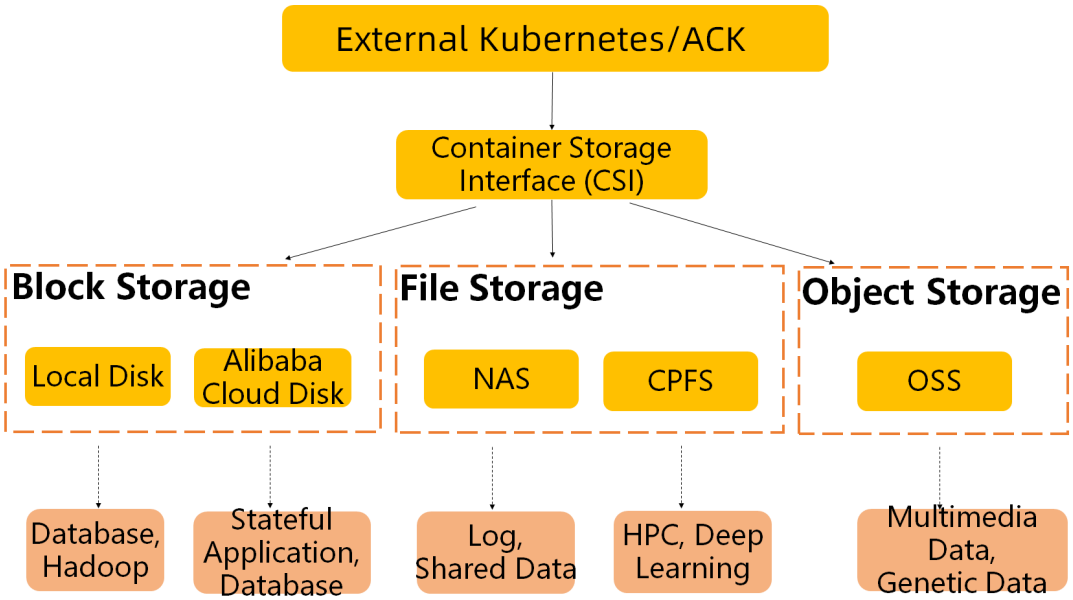
```
spec:
 imagePullSecrets:
- nameregsecretey
```

8.Storage management-CSI

8.1. Storage overview

Container Service for Kubernetes (ACK) provides container storage features based on the Kubernetes storage system. The storage features are integrated with Alibaba Cloud storage services and are fully compatible with native storage objects of Kubernetes, such as EmptyDir, HostPath, Secrets, and ConfigMaps. ACK provides the Container Storage Interface (CSI) plug-in based on the open source CSI. You must deploy the CSI plug-in before you can use Alibaba Cloud storage services in ACK clusters. This topic describes the overview, features, and limits of the CSI plug-in, and the permissions that are required to use the CSI plug-in.

Container storage architecture



ACK allows you to configure storage services to be automatically mounted on pods. The storage services include Alibaba Cloud disks, Apsara File Storage NAS (NAS), Object Storage Service (OSS), Cloud Paralleled File System (CPFS), and local volumes. The following table describes the features and use scenarios of the storage services.

Alibaba Cloud storage service	Statically provisioned volume	Dynamically provisioned volume	Deployed by default	Feature	Scenario

Alibaba Cloud storage service	Statically provisioned volume	Dynamically provisioned volume	Deployed by default	Feature	Scenario
Disk	Supported	Supported	Yes	Non-shared storage. A disk can be mounted on only one node.	• High I/O and low latency Disks are block storage devices and are suitable for use in scenarios that require high I/O performance and low latency. For example, databases and middleware services. • Non-data sharing A disk can be used to provision storage for only one pod. You can use disks as volumes in scenarios that do not require data sharing. For more information, see Overview.
NAS	Supported	Supported	Yes	Shared storage that provides high performance and high throughput.	• Data sharing NAS file systems allow multiple pods to access the same data. We recommend that you use NAS file systems if data needs to be shared. • Big data analysis NAS file systems provide high throughput and meet the requirement of shared storage access when large numbers of jobs are involved. • Web applications NAS file systems can provision storage for web applications and content management systems. • Log storage We recommend that you use NAS file systems as volumes to store log data. For more information, see Overview.

Alibaba Cloud storage service	Statically provisioned volume	Dynamically provisioned volume	Deployed by default	Feature	Scenario
OSS	Supported	Not supported	Yes	Shared storage that supports file systems in the user space.	• Read-only media files such as video and image files You can use OSS buckets as volumes to read the preceding types of files. • Read-only configuration files of websites and applications ossfs provides limited network performance and can be used to read small files. **Note** OSS buckets are mounted by using ossfs, which is implemented as a filesystem in the user space (FUSE). The write performance is limited when you use OSS buckets as volumes. We recommend that you use other storage media as volumes in scenarios that require high write performance. For more information, see Overview.
CPFS	Supported	Supported	No	Shared storage that features high performance and high bandwidth.	• Genomics computing and big data analysis CPFS provides high throughput and meets the requirement of high performance in large-scale clusters. • Data cache access You can download data from low-speed storage to CPFS volumes. This way, applications can access the data at a high speed. For more information, see Static volumes and Dynamic volumes.

Note You can use the CSI plug-in to mount storage as statically provisioned volumes and dynamically provisioned volumes in ACK clusters. To mount storage as a statically provisioned volume, you must create a persistent volume (PV) and a persistent volume claim (PVC). When large numbers of PVs and PVCs are required, you can mount storage as dynamically provisioned volumes. Definitions of PV and PVC:

PV

A PV is a piece of storage in the cluster. A PV has a lifecycle that is independent of the pod that uses the PV. Different types of PV can be created based on different StorageClasses.

PVC

A PVC is a request for storage in the cluster. PVs are node resources consumed by pods. PVCs are claims that consume PV resources. When PVs are insufficient, PVCs can dynamically provision PVs.

Container storage features

The following table describes the storage features supported by different types of ACK cluster.

Storage type	Feature	ACK cluster (Linux-based)	ASK cluster	Registered cluster (hybrid cloud or multicloud)	ACK@Edge	ACK cluster (Windows-based)	Dedicated ACK cluster	Sandboxed container
Block storage	Mount and unmount disks	✔	✔	✘	✘	✔	✔	✔
	Resize	✔	✘	✘	✘	✘	✔	✘
	Snapshots	✔	✔	✘	✘	✘	✔	✔
	Container I/O monitoring	✔	✘	✘	✘	✘	✔	✘
	File systems	XFS, ext4, and dBFS are supported.	XFS and ext4 are supported.	✘	✘	NTFS is supported.	XFS and ext4 are supported.	XFS and ext4 are supported.
	Block and bare devices	✔	✘	✘	✘	✘	✔	✘
	Data restoration from snapshots	✔	✔	✘	✘	✘	✔	✔
	Disk queue settings	✔	✘	✘	✘	✘	✔	✘

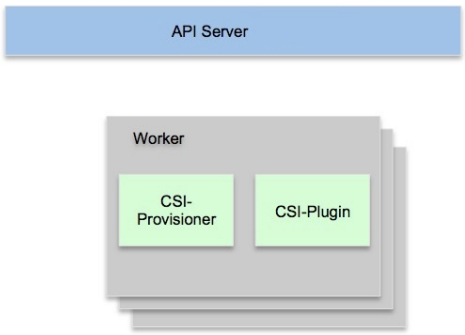
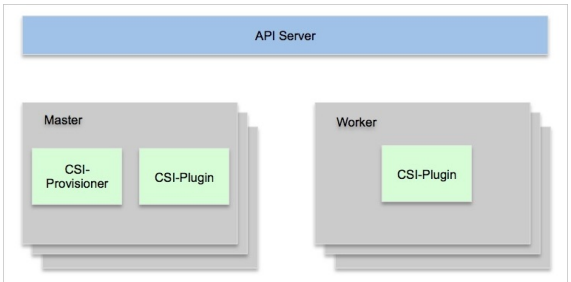
Storage type	Feature	ACK cluster (Linux-based)	ASK cluster	Registered cluster (hybrid cloud or multicloud)	ACK@Edge	ACK cluster (Windows-based)	Dedicated ACK cluster	Sandboxed container
	Customer Managed Key (CMK) and Bring Your Own Key (BYOK)	✔	✔	✘	✘	✔	✔	✔
	Multi-zone	✔	✘	✘	✘	✔	✔	✔
	Custom labels	✔	✔	✘	✘	✔	✔	✔
	Cross-host migration	✔	✔	✘	✘	✔	✔	✔
File storage	Create, mount, and unmount NAS file systems	✔	✔	✔	✔	✘	✔	✔
	Mount and unmount Samba file systems	✘	✘	✔	✘	✔	✘	✘
	CPFS	✔	✘	✔	✘	✘	✘	✘
	NAS recycle bin	✘	✘	✘	✘	✘	✘	✘
	Subdirectories of dynamically provisioned volumes	✔	✘	✔	✔	✔	✔	✔
	CMK (Extreme NAS)	✔	✘	✘	✘	✘	✔	✔
Object storage	Mount and unmount OSS buckets	✔	✔	✔	✔	✘	✔	✔
	BYOK	✔	✔	✔	✔	✘	✔	✔
	Linux Volume Manager (LVM)-managed block storage	✔	✘	✔	✔	✘	✔	✘

Storage type	Feature	ACK cluster (Linux-based)	ASK cluster	Registered cluster (hybrid cloud or multicloud)	ACK@Edge	ACK cluster (Windows-based)	Dedicated ACK cluster	Sandboxed container
Local storage	Automated volume groups	✔	✘	✔	✔	✘	✔	✘
	Node capacity scheduling	✔	✘	✔	✔	✘	✔	✘
	PMEM Direct Mem	✔	✘	✘	✘	✘	✘	✘
	LVM-managed persistent memory (PMEM)	✔	✘	✘	✘	✘	✘	✘

CSI deployment architectures

The CSI plug-in consists of two parts: CSI-Plugin and CSI-Provisioner. The following figure shows the deployment architectures of the CSI plug-in in a managed Kubernetes cluster and a dedicated Kubernetes cluster.

> **Note** The CSI plug-in is automatically installed in managed and dedicated Kubernetes clusters. If you use a serverless Kubernetes cluster or an edge Kubernetes cluster, you must manually install the CSI plug-in.

Managed Kubernetes cluster	Dedicated Kubernetes cluster
 In managed Kubernetes clusters, CSI-Provisioner and CSI-Plugin are deployed on worker nodes.	 In dedicated Kubernetes clusters, CSI-Provisioner is deployed on master nodes. CSI-Plugin is automatically deployed as DaemonSets on master and worker nodes.

Permissions required to use CSI

Before you can use the CSI plug-in to mount, unmount, create, and delete volumes, you must grant the plug-in the permissions to access other cloud resources. You can use an AccessKey pair or a Resource Access Management (RAM) role to grant permissions to the CSI plug-in. The default method is to use a RAM role. The following table describes the two authorization methods.

Use an AccessKey pair	Use a RAM role
• You can specify an AccessKey pair in the deployment template of the CSI plug-in. • You can also create a Secret to pass an AccessKey pair as environment variables.	The CSI plug-in uses the RAM role **AliyunCSManagedCsiRole** to access resources of other cloud services. For more information, see AliyunCSManagedCsiRole. For more information about how to grant permissions to a RAM role, see Grant permissions to a RAM role. • Managed clusters The permission token of the RAM role used by the CSI plug-in is stored in a Secret named **addon.csi.token**. To grant permissions to the CSI plug-in and allow the plug-in to call API operations, you need only to mount the Secret to the plug-in. • Dedicated clusters The CSI plug-in assumes the RAM role of the Elastic Compute Service (ECS) node that hosts the pod of the plug-in.

CSI limits

When you use the CSI plug-in in ACK clusters, take note of the limits of the CSI plug-in and Alibaba Cloud storage services.

- **Limits of Alibaba Cloud storage services**

Alibaba Cloud storage service	Limits
Disk	◦ You can mount up to 15 disks as volumes on a node. ◦ You can provision a disk to only one pod as a volume. ◦ You cannot mount disks of all types on all ECS instances. For more information, see Instance families. ◦ You cannot mount or unmount subscription disks as volumes. ◦ You can mount a disk only on an ECS instance that is in the same zone as the disk. ◦ We recommend that you create StatefulSets instead of Deployments to use volumes that are created from disks. **Note** Deployments are used to create stateless applications. When a pod is restarted, the start time of the new pod may overlap the end time of the old pod. If multiple pods are created for a Deployment, no dedicated volume is provisioned for each pod. ◦ The minimum capacity of each volume is 20 GiB.

Alibaba Cloud storage service	Limits
NAS	◦ You can mount a NAS file system only on an ECS instance that is deployed in the same virtual private cloud (VPC) as the NAS file system. ◦ The number of NAS file systems that you can create is subject to a quota limit. To request a quota increase, Submit a ticket to the NAS team.
OSS	We recommend that you do not perform data write operations on volumes that are created from OSS buckets. Use other storage media for data write operations. **Note** OSS buckets are mounted by using ossfs, which is implemented as a filesystem in the user space (FUSE). The write performance is limited when you use OSS buckets as volumes. We recommend that you use other storage media as volumes in scenarios that require high write performance.
CPFS	◦ The CPFS driver is highly dependent on the OS kernel. After you deploy the CPFS environment, do not upgrade the OS kernel. ◦ You can install the CPFS driver but cannot upgrade the CPFS driver. **Note** The CSI-CPFS plug-in is a Kubernetes CSI component that mounts CPFS file systems as volumes on pods for the use of applications. The CPFS driver is a client driver that implements the CPFS protocol at the kernel layer. Relationship between the CSI-CPFS plug-in and the CPFS driver: ◦ If the CPFS driver is not installed on a node, the CPFS driver is automatically installed when you deploy the CSI-CPFS plug-in. ◦ If the CPFS driver is installed on a node, the CPFS driver is not installed or upgraded when you deploy the CSI-CPFS plug-in.

- **Limits of the CSI plug-in**
  The CSI plug-in is an open source plug-in for ACK clusters. In other types of clusters, such as clusters deployed in third-party clouds and self-managed clusters on Alibaba Cloud, you cannot directly use the CSI plug-in for reasons such as cluster configurations, permission management, and network differences. If you want to use the CSI plug-in in these types of clusters, you must modify the cluster configurations based on the source code. For more information, see alibaba-cloud-csi-driver.

Kubernetes version requirements

To use the CSI plug-in in an ACK cluster, the Kubernetes version of the cluster must be 1.14 or later. Besides, the kubelet parameter `--enable-controller-attach-detach` must be set to `true`.

Installation and upgrade of the CSI plug-in

For more information about how to install and upgrade the CSI plug-in, see Install and upgrade the CSI plug-in.

Differences between the CSI and FlexVolume plug-ins

Plug-in	Feature	References

Plug-in	Feature	References
Flexvolume	FlexVolume is a traditional mechanism to extend Kubernetes storage systems developed by the Kubernetes community. ACK supports FlexVolume. FlexVolume consists of the following parts: • FlexVolume: allows you to mount and unmount volumes. By default, ACK allows you to mount the following types of storage media: disks, NAS file systems, and OSS buckets. • Disk-Controller: automatically mounts disks as volumes. • Nas-Controller: automatically mounts NAS file systems as volumes.	For more information about FlexVolume, see Overview. For more information about how to upgrade FlexVolume, see Manage system components.
CSI	The Kubernetes community recommends the CSI plug-in. The CSI plug-in provided by ACK is compatible with the features of the community version. CSI consists of the following two parts: • CSI-Plugin: allows you to mount and unmount volumes. By default, ACK allows you to mount the following types of storage media: disks, NAS file systems, and OSS buckets. • CSI-Provisioner: automatically mounts disks and NAS file systems as volumes.	For more information about CSI, see Overview and alibaba-cloud-csi-driver.

Note

- You must select a plug-in when you create an ACK cluster.
- You cannot use CSI and FlexVolume in the same cluster.
- You cannot change the plug-in from FlexVolume to CSI for a cluster.

Recommendation

- For new ACK clusters, we recommend that you use CSI. The ACK technical team will continuously upgrade CSI to support more features of the CSI community version.
- For existing clusters, we recommend that you use the plug-in that is already installed. The ACK technical team will continue its support for FlexVolume.

How to check the storage plug-in used in a cluster

- Method 1: Check node annotations by using the console
  i. Log on to the ACK console.
  ii. In the left-side navigation pane, click **Clusters**.
  iii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
  iv. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
  v. Select a node and click **More > Details** in the **Actions** column.
  vi. In the **Overview** section, check **Annotations**.
  If `volumes.kubernetes.io/controller-managed-attach-detach: true` is displayed, the cluster uses the CSI plug-in. If `volumes.kubernetes.io/controller-managed-attach-detach: true` is not displayed, the cluster uses the FlexVolume plug-in.

- Method 2: Check kubelet parameters
  Run the following command to check kubelet parameters:

  ```
  ps -ef | grep kubelet
  ```

  Expected output:

  ```
  --enable-controller-attach-detach=true
  ```

  If the value of `--enable-controller-attach-detach` is `true`, the cluster uses the CSI plug-in. If the value of `--enable-controller-attach-detach` is `false`, the cluster uses the FlexVolume plug-in.

Related information

- Storage basics
- Overview
- Overview
- Overview
- Static volumes
- Dynamic volumes
-
-

8.2. Storage basics

Container Service for Kubernetes (ACK) uses the Kubernetes orchestration system as a management platform for clusters, applications, storage, networks and other modules. This topic describes the basics about container storage in ACK. You can use this topic to help you understand the basics and principles of storage modules when you use the storage features provided by ACK.

Volumes

On-disk files in a container are ephemeral. The Kubernetes volume abstraction solves the problem that files are lost when the container restarts. A volume is a data transfer channel between a pod and an external storage device. A volume also enables data sharing between containers in a pod, between pods, and between a pod and the external environment.

A volume defines the details of external storage and is embedded in a pod. A volume is essentially a resource mapping to external storage in the Kubernetes system. When a workload requires external storage, the system queries related information in the volume and mounts external storage.

> **Note** A volume has the same lifecycle as the pod that uses the volume. When the pod is deleted, the volume is deleted at the same time. Whether the data in the volume is retained depends on the volume type.

Kubernetes provides a wide variety of volume types. ACK supports the following commonly used volume types:

Volume	Description
Local storage	You can mount local storage as volumes, such as HostPath and emptyDir volumes. These volumes store data on specific nodes of the cluster and do not drift with applications. The stored data becomes unavailable when the nodes are down.

Volume	Description
Network storage	You can mount network storage as volumes, such as Ceph, GlusterFS, NFS, and iSCSI volumes. These volumes store data by using remote storage services. When you use these volumes, you must mount the storage services locally.
Secrets and ConfigMaps	Secrets and ConfigMaps are special volumes that store the object information about a cluster. Object data is mounted as volumes to nodes for use by applications.
PVC	Persistent volume claims (PVCs) provide a mechanism for defining volumes. A PVC abstracts a volume into a pod-independent object. The storage information that is defined for or associated with this object is stored in a volume and mounted for use by Kubernetes workloads.

Note Container Storage Interface (CSI) and FlexVolume are two extensions of volumes. Each extension can be divided into different StorageClasses. For more information about CSI, see Storage overview.

Notes about volumes

- A pod can mount multiple volumes.
- A pod can mount multiple types of volume.
- A volume that is mounted on a pod can be shared among the containers in the pod.
- We recommend that you create persistent volumes (PVs) and PVCs to mount volumes in Kubernetes.
- We recommend that you do not mount an excess number of volumes on a pod.

PVs and PVCs

Not all Kubernetes volumes are persistent. Container storage requires a remote storage service to enable data persistence. To do this, Kubernetes introduces two resource objects, PV and PVC, which abstract details of how storage is implemented from how it is consumed, and decouple the responsibility of the user from the system administrator. The details of PV and PVC are:

PV

PV is the abbreviation for persistent volume. PVs are a specific type of volume in Kubernetes. A PV object defines a specific StorageClass and a set of volume parameters. All information about the target storage service is stored in a PV object. Kubernetes references the information stored in the PV object to mount volumes.

Note PVs are used at the cluster level instead of the node level. A PV has its own lifecycle, which is independent of the pod lifecycle.

PVC

PVC is the abbreviation for persistent volume claim. PVCs are a type of abstract volume in Kubernetes and represents the data volume of a specific StorageClass. PVCs are designed to separate storage from application orchestration. A PVC object abstracts storage details and implements storage orchestration. This makes storage volume objects independent of application orchestration in Kubernetes and decouples applications from storage at the orchestration layer.

Notes about PVs and PVCs

- The binding of PVs and PVCs

One PVC object is bound to only one PV object. One PV object cannot be bound to multiple PVC objects, and one PVC object cannot be bound to multiple PV objects. To configure storage for an application, you must declare a PVC object. Kubernetes selects the PV object that best fits the PVC object and binds them together. PVCs are a type of storage object used by applications and belong to the application domain. PVs are a type of storage object that belongs to the storage domain.
A PVC object must be bound to a PV object before it can be consumed by a pod. The process of binding a PVC object to a PV object is the process of PV consumption. Only a PV object that meets the following requirements can be bound to a PVC object:

- VolumeMode: The PV object to be consumed must be in the same volume mode as the PVC object.
- AccessMode: The PV object to be consumed must be in the same access mode as the PVC object.
- StorageClassName: If this parameter is defined for a PVC object, only a PV object that has the corresponding parameters defined can be bound to this PVC object.
- LabelSelector: The appropriate PV object is selected from a PV list based on label matching.
- size: The PV object to be consumed must have a storage capacity that is no less than that of the PVC object.

- **The size fields in PV and PVC definitions**
  The size fields in PV and PVC definitions have the following uses:
  - The system determines whether a PV object and a PVC object can be bound based on the size parameters.
  - To dynamically create a PV object based on a PVC object and a StorageClass, some storage types determine the capacities of the PV object and backend storage based on the size of the PVC object.
  - For storage types that support resize operations, the size of the PVC object is used as the capacities of the PV object and backend storage after the scale-out.

  The size values of a PVC object and a PV object are used as configuration parameters when the system performs operations on the PVC object and PV object.
  When data is written to the volume, the size fields of the PVC object and PV object are not referenced. Instead, write operations depend on the actual capacity of the underlying storage medium.

How to use volumes

Volumes are commonly used in the following ways:

- **Static volumes**
  Static volumes are PV objects created by administrators. All volumes can be created as static volumes. The cluster administrator analyzes the storage needs of the cluster and pre-allocates storage media, such as disks and NAS file systems. The administrator also creates PV objects to be consumed by PVC objects. If PVC needs are defined in workloads, Kubernetes binds PVC and PV objects based on relevant rules. This allows applications to access storage services.
- **Dynamic volumes**
  Dynamic volumes are PV objects automatically created by the storage plug-in.
  The cluster administrator configures a backend storage pool and creates a StorageClass. When a PVC object needs to consume a PV object, the storage plug-in dynamically creates a PV object based on the PVC needs and the details of the StorageClass. The definition of StorageClass is:

- **StorageClass**
  When you declare a PVC object, you can add the StorageClassName field to this PVC object. This allows the Provisioner plug-in to create a suitable PV object based on the definition of StorageClassName when no PV object in the cluster fits the declared PVC object. This creates a dynamic volume.
  The dynamic volume is created by the Provisioner plug-in and bound with the PVC object based on StorageClassName.
  A StorageClass is a template used to create a PV object. When a PVC object triggers the automatic PV creation process, a PV object is created based on the content of a StorageClass. The following example shows the content of a StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: alicloud-disk-topology
parameters:
 type: cloud_ssd
provisioner: diskplugin.csi.alibabacloud.com
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

Parameter	Description
reclaimPolicy	Specifies the value of the persistentVolumeReclaimPolicy field used to create a PV object. Valid values: *Delete* and *Retain*. ■ *Delete* specifies that a dynamically created PV object is automatically released when the bound PVC object is released. ■ *Retain* specifies that a PV object is dynamically created, but must be released by the administrator.
allowVolumeExpansion	Specifies whether the PV object created based on the current StorageClass performs dynamic scale-out. Default value: `false`. This parameter is used only to enable or disable dynamic scale-out. Whether to enable this feature is determined by the underlying storage plug-in.
volumeBindingMode	Specifies the time when PV objects are dynamically created. Valid values: *Immediate* (immediate creation) and *WaitForFirstConsumer* (delayed creation).

- **Delayed binding**
  Certain types of storage, such as Alibaba Cloud disks, impose limits on the mount attribute. For example, volumes can be mounted to only nodes in the same zone as the volumes. This type of volume encounters the following problems:
  - A volume is created in Zone A, but Zone A has no available node resources. Therefore, the created volume cannot be mounted to a pod upon startup.
  - When cluster administrators plan PVC and PV objects, they cannot determine in which zones they can create multiple PV objects for backup.

  The StorageClass template provides the volumeBindingMode field to fix the preceding problems. When volumeBindingMode is set to *WaitForFirstConsumer*, the storage plug-in delays volume creation when it receives the PVC pending state. Instead, the storage plug-in creates a volume only after the PVC object is consumed by a pod.

  > **Note**
  > The process of delayed binding:
  > The Provisioner plug-in delays volume creation when it receives the PVC pending state. Instead, the plug-in creates a volume only after the PVC object is consumed by a pod. If a pod consumes the PVC object and the scheduler determines that the PVC object has delayed binding enabled, the PV scheduling process continues. The scheduler **patches** the scheduling result to the metadata of the PVC object. When the Provisioner plug-in determines that scheduling information has been written to the PVC object, it retrieves location information such as the zone and node based on the scheduling information to create a volume. Then, the plug-in triggers the PV creation process.
  > We recommend that you use the delayed binding feature when you create dynamic volumes in a multi-zone cluster. The preceding configuration process is supported by ACK.

Advantages of dynamic volumes

- Dynamic volumes allow Kubernetes to implement automatic PV lifecycle management. PV objects are created and deleted by the Provisioner plug-in.
- PV objects can be automatically created, which simplifies configuration and reduces the workload of system administrators.
- Dynamic volumes maintain consistency between the PVC-required storage capacity and the PV capacity that is configured by the Provisioner plug-in. This optimizes storage capacity planning.

- **Mount SubPath volumes**
  Kubernetes provides the `volumeMounts.subPath` property, which can be used to specify a subpath inside the referenced volume instead of its root.

  > **Note** We recommend that you do not use SubPath volumes because the mount and unmount of SubPath volumes may cause errors in some Kubernetes versions. You can use the Container Storage Interface (CSI) plug-in to mount subdirectories, such as those of NAS file systems and OSS buckets.

Quota limits

- Certain types of storage impose quota limits on the number of volumes that can mounted on a node. For example, you can mount up to 16 disks as volumes on a node. ACK provides scheduling policies that meet quota limits. When the number of volumes that are mounted on a node reaches the quota, the scheduler no longer schedules the specified type of volume to this node.
- The CSI plug-in provided by ACK allows you to use the MAX_VOLUMES_PERNODE field to configure the number of volumes per node. For more information about the CSI plug-in, see Storage overview.

References

- For more information about Kubernetes storage, see Storage.
- For more information about the CSI plug-in, see Storage overview.

8.3. Install and upgrade the CSI plug-in

The CSI plug-in consists of CSI-Plugin and CSI-Provisioner. This topic describes how to install and upgrade the CSI plug-in in a Container Service for Kubernetes (ACK) cluster.

Prerequisites

- A cluster of ACK later than 1.14 is created, and the CSI plug-in is specified as the volume plug-in of the cluster. For more information, see 创建Kubernetes托管版集群.
- You are connected to the cluster by using kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

Install CSI-Plugin and CSI-Provisioner

If you do not specify FlexVolume as the volume plug-in when you create a managed Kubernetes cluster or a dedicated Kubernetes cluster, the system installs CSI-Plugin and CSI-Provisioner by default.

Verify the installation

Check whether CSI-Plugin and CSI-Provisioner are installed in the cluster.

- Run the following command to check whether CSI-Plugin is installed in the cluster:

```
kubectl get pod -n kube-system | grep csi-plugin
```

- Run the following command to check whether CSI-Provisioner is installed in the cluster:

```
kubectl get pod -n kube-system | grep csi-provisioner
```

Upgrade CSI-Plugin and CSI-Provisioner

You can upgrade CSI-Plugin and CSI-Provisioner in the ACK console.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. Click the **Storage** tab, find **csi-plugin** and **csi-provisioner**, and click **Upgrade**.
6. In the **Note** message, confirm the versions of the plug-ins and click **OK**.
   After the plug-ins are upgraded, the system prompts that the upgrades are completed and the current versions of the plug-ins are displayed.

If the plug-ins fail to be upgraded in the console or the plug-ins fail to pass the precheck, you can perform the following operations accordingly:

- CSI-Plugin fails to pass the precheck.
  - If volumes that use disks, Apsara File Storage NAS (NAS) file systems, or Object Storage Service (OSS) buckets are not provisioned in the cluster, you must manually upgrade CSI-Plugin. For more information, see Upgrade CSI-Plugin.
  - If volumes that use disks, NAS file systems, or OSS buckets are provisioned in the cluster, and the cluster is created in a staging environment, you must manually upgrade CSI-Plugin. For more information, see Upgrade CSI-Plugin.

  - If volumes that use disks, NAS file systems, or OSS buckets are provisioned in the cluster, and business critical data is stored in the volumes, Submit a ticket to request technical support.

- CSI-Plugin passes the precheck but fails to be upgraded.
  Check whether the nodes in the cluster are in the Ready state. If CSI-Plugin is installed on a node that is in the NotReady state, you must fix the state of the node.
  If you cannot identify the cause of the failure, Submit a ticket to request technical support.
- CSI-Plugin is displayed in the console but CSI-Provisioner is not displayed.
  CSI-Provisioner is deployed by using a StatefulSet. In this case, Submit a ticket to request technical support.
- CSI-Provisioner fails to pass the precheck.
  - If no volumes that use disks or NAS file systems are dynamically provisioned by using StorageClasses in the cluster, you must manually upgrade CSI-Provisioner. For more information, see Upgrade CSI-Provisioner.
  - If volumes that use disks or NAS file systems are dynamically provisioned by using StorageClasses in the cluster, and the cluster is created in a staging environment, you must manually upgrade CSI-Provisioner. For more information, see Upgrade CSI-Provisioner.
  - If volumes that use disks or NAS file systems are dynamically provisioned by using StorageClasses in the cluster, and business critical data is stored in the volumes, Submit a ticket to request technical support.
- CSI-Provisioner passes the precheck but fails to be upgraded. In this case, Submit a ticket to request technical support.

8.4. Disk volumes

8.4.1. Overview

You can create volumes to mount Alibaba Cloud disks to a Container Service for Kubernetes (ACK) cluster. You can use the Contain Storage Interface (CSI) plug-in provided by Alibaba Cloud to mount disks by creating persistent volumes (PVs) and persistent volume claims (PVCs). A PV can be statically or dynamically provisioned. This topic describes the limits on and requirements of disk volumes.

Usage notes

- A disk cannot be shared. If you use a disk by creating a pair of PV and PVC, the disk can be mounted to only one pod.
- We recommend that you mount a disk to an application deployed by using a StatefulSet. If you mount a disk to an application deployed by using a Deployment, you must set the number of pod replicas to one. In addition, you cannot prioritize pods when you mount or unmount disks. We recommend that you do not mount a disk to an application deployed by using a Deployment.
- Before you can use a statically provisioned disk volume, you must create the disk and obtain the disk ID. For more information, see Create a disk.

  > **Note** Your disk must meet the following requirements:
  > - An ultra disk must have a minimum capacity of 20 GiB.
  > - A standard SSD must have a minimum capacity of 20 GiB.
  > - An enhanced SSD (ESSD) must have a minimum capacity of 20 GiB.

- You can mount disks to only the nodes that are deployed in the same zone as the disks. Therefore, when you create a disk, select the zone of the pod to which you want to mount the disk.
- The type of disk must match the Elastic Compute Service (ECS) instance types that are used in your cluster

before you can mount a disk. For more information about the matching rules between disk types and ECS instance types, see Instance families.

- Only pay-as-you-go disks can be mounted. If you change the billing method of an Elastic Compute Service (ECS) instance in the cluster from pay-as-you-go to subscription, you cannot change the billing method of its disks to subscription. Otherwise, the disks cannot be mounted to the cluster.

StorageClass

StorageClass

ACK clusters support the following types of StorageClass:

- alicloud-disk-efficiency: ultra disk.
- alicloud-disk-ssd: standard SSD.
- alicloud-disk-essd: ESSD.
- alicloud-disk-available: a high-availability mode. In this mode, the system attempts to create a standard SSD in priority. If the standard SSD resources are exhausted, the system attempts to create an ultra disk.

  > **Notice** For alicloud-csi-provisioner earlier than V1.14.8.39-0d749258-aliyun, the system attempts to create a disk in the order of enhanced SSD, standard SSD, and ultra disk.

- alicloud-disk-topology: creates a disk in WaitForFirstConsumer mode.

The first four types of StorageClass are suitable for single-zone clusters. The last type of StorageClass is suitable for multi-zone clusters.

When you configure a StorageClass, the following rules determine the zone where the disk is created:

- Specify volumeBindingMode: WaitForFirstConsumer in the StorageClass configurations to create a disk in the zone where the pod that uses the disk volume is deployed.
- Specify volumeBindingMode: Immediate and specify a zone by setting the zoneId parameter in the StorageClass configurations to create a disk in the specified zone.
- Specify volumeBindingMode: Immediate and specify multiple zones by setting the zoneId parameter in the StorageClass configurations to create a disk in one of the specified zones.
- Specify volumeBindingMode: Immediate without setting the zoneId parameter in the StorageClass configurations to create a disk in the zone where csi-provisioner is deployed.

If your cluster is deployed across zones, we recommend that you set volumeBindingMode: WaitForFirstConsumer in the StorageClass configurations. You can create a StorageClass based on the type of disk that is required.

Default StorageClass

Kubernetes provides the default StorageClass feature. If a PVC does not specify a StorageClass, the default StorageClass is used to provision a PV for the PVC. For more information, see Default StorageClass.

> **Note**
>
> - The default StorageClass applies to all PVCs. Exercise caution if your cluster has PVCs for multiple storage media. For example, the default StorageClass may create a cloud disk as the PV for a PVC that defines an Apsara File Storage NAS (NAS) file system. Therefore, ACK clusters do not provide default StorageClasses. If you want to configure a default StorageClass, perform the following steps.
> - You can configure only one default StorageClass for each cluster. If you configure more than one default StorageClass for a cluster, all default StorageClasses become invalid.

1. Configure a default StorageClass.

Run the following command to set alicloud-disk-ssd as a default StorageClass:

```
kubectl patch storageclass alicloud-disk-ssd -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

After the default StorageClass is configured, alicloud-disk-ssd is marked as (default)

```
kubectl get sc
```

The following output is returned:

```
NAME                 PROVISIONER              AGE
alicloud-disk-ssd (default)  diskplugin.csi.alibabacloud.com  96m
```

2. Use the default StroageClass.
   i. The following code provides an example on how to create a PVC without specifying a StorageClass:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: disk-pvc
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 20Gi
```

   The cluster automatically creates a cloud disk as the PV based on the default StorageClass alicloud-disk-ssd.

```
kubectl get pvc
```

   The following output is returned:

```
NAME    STATUS  VOLUME          CAPACITY  ACCESS MODES  STORAGECLASS     AGE
disk-pvc  Bound  d-bp18pbai447qverm3ttq  20Gi    RWO      alicloud-disk-ssd  49s
```

What's next

You can also run the following command to disable the default StorageClass:

```
kubectl patch storageclass alicloud-disk-ssd -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

8.4.2. Use a statically provisioned disk volume

Alibaba Cloud disks are block-level storage resources for Elastic Compute Service (ECS). Alibaba Cloud disks provide low latency, high performance, high durability, and high reliability. Container Service for Kubernetes (ACK) allows you to use the Container Storage Interface (CSI) plug-in to statically and dynamically provision disk volumes. This topic describes how to use CSI to mount a statically provisioned disk volume and how to enable persistent storage by using a statically provisioned disk volume.

Prerequisites

- An ACK cluster is created and the CSI plug-in is deployed in the cluster. For more information, see 创建Kubernetes托管版集群.

- A pay-as-you-go disk is created. The disk ID is `d-wz92s6d95go6ki9x****`. For more information, see Create a disk.

  > **Note** Your disk must meet the following requirements:
  > - An ultra disk must have a minimum capacity of 20 GiB.
  > - A standard SSD must have a minimum capacity of 20 GiB.
  > - An enhanced SSD (ESSD) must have a minimum capacity of 20 GiB.

- Your machine is connected to the cluster by using kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

Context

Scenarios:

- You want to create applications that require high disk I/O and do not require data sharing. The applications can use storage services such as MySQL and Redis.
- You want to collect log data at high speeds.
- You want to persist data in a way that is independent of the pod lifecycle.

To mount a disk as a statically provisioned volume, make sure that you have purchased a disk.

Manually create a persistent volume (PV) and a persistent volume claim (PVC) that are used to statically provision a disk.

Limits

- Alibaba Cloud disks cannot be shared. A disk can be mounted to only one pod.
- A disk can be mounted to only a node that is deployed in the same zone as the disk.

Use a statically provisioned disk volume in the ACK console

Step 1: Create a PV.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.
5. In the upper-right corner of the **Persistent Volumes** page, click **Create**.
6. In the **Create PV** dialog box, set the following parameters.

Parameter	Description
PV Type	You can select Cloud Disk, NAS, or OSS. In this example, Cloud Disk is selected.
Volume Plug-in	You can select Flexvolume or CSI. In this example, CSI is selected.
Access Mode	By default, this parameter is set to ReadWriteOnce.
Disk ID	Select a mountable disk that is deployed in the same region and zone as your cluster.
File System Type	Select the file system of the disk. Valid values: **ext4**, **ext3**, **xfs**, and **vfat**. Default value: **ext4**.

Parameter	Description
Label	Add labels to the PV.

7. Click **Create**.

Step 2: Create a PVC.

1. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volume Claims**.
2. In the upper-right corner of the **Persistent Volume Claims** page, click **Create**.
3. In the **Create PVC** dialog box, set the following parameters:

Parameter	Description
PVC Type	You can select Cloud Disk, NAS, or OSS. In this example, Cloud Disk is selected.
Name	The name of the PVC. The name must be unique in the namespace.
Allocation Mode	In this example, Existing Volumes is selected. **Note** If no PV is created, you can set **Allocation Mode** to **Create Volume**, and set other parameters to create a PV. For more information, see . For more information, see Create a PV.
Existing Volumes	Click **Select PV**. In the dialog box that appears, find the PV that you want to use and click **Select** in the Actions column.
Capacity	The capacity claimed by the PVC. **Note** The capacity claimed by the PVC cannot exceed the capacity of the PV.
Access Mode	By default, this parameter is set to ReadWriteOnce.

4. Click **Create**.

   After the PVC is created, you can find the PVC in the list of PVCs. The PVC is bound to the specified PV.

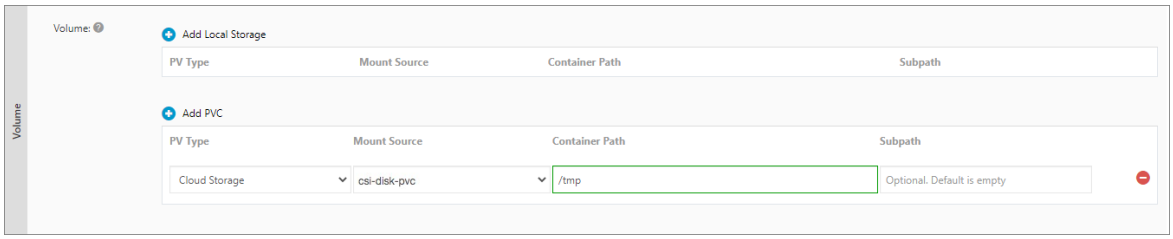
Step 3: Create an application

1. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
2. In the upper-right corner of the **Deployments** page, click **Create from Image**.
3. Set the parameters that are used to create an application.

   This example shows how to set the volume parameters. For more information about other parameters, see Create a stateless application by using a Deployment.
   You can add local volumes and cloud volumes.

   - **Add Local Storage**: You can select HostPath, ConfigMap, Secret, or EmptyDir from the PV Type drop-down list. Then, set the Mount Source and Container Path parameters to mount the volume to a container path. For more information, see Volumes.
   - **Add PVC**: You can add cloud volumes.

Mount the disk volume that is created in this example to the */tmp* path of the container. After the disk volume is mounted, container data that is generated in the /tmp path is stored in the disk volume.



4. Set other parameters and click **Create**.

   After the application is created, you can use the disk volume to store application data.

Mount a statically provisioned disk volume by using kubectl

Step 1: Create a PV

1. Create a PV by using the following *pv-static.yaml* file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: csi-pv
 labels:
  alicloud-pvname: static-disk-pv
spec:
 capacity:
  storage: 25Gi
 accessModes:
  - ReadWriteOnce
 persistentVolumeReclaimPolicy: Retain
 csi:
  driver: diskplugin.csi.alibabacloud.com
  volumeHandle: "<your-disk-id>"
 nodeAffinity:
  required:
   nodeSelectorTerms:
   - matchExpressions:
     - key: topology.diskplugin.csi.alibabacloud.com/zone
       operator: In
       values:
       - "<your-node-zone-id>"
```

Parameter	Description
name	The name of the PV.
labels	The labels that are added to the PV.
storage	The available storage of the disk.
accessModes	The access mode of the PV.
persistentVolumeReclaimPolicy	The policy for reclaiming the PV.

Parameter	Description
driver	The type of driver. This parameter is set to diskplugin.csi.alibabacloud.com. This value indicates that the Alibaba Cloud CSI plug-in is used.
volumeHandle	The ID of the disk.
nodeAffinity	Information about the zone to which the PV and PVC belong. You can set this parameter to specify the zone to which the pod that uses the PV and PVC is scheduled.

2. Run the following command to create a PV:

```
kubectl create -f pv-static.yaml
```

Log on to the ACK console. In the left-side navigation pane, click **Clusters**. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster, or click **Details** in the **Actions** column. The details page of the cluster appears. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**. You can view the created PV on the **Persistent Volumes** page.

Step 2: Create a PVC

1. Use the following template to create a *pvc-static.yaml* file:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: csi-pvc
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 25Gi
 selector:
  matchLabels:
   alicloud-pvname: static-disk-pv
```

Parameter	Description
name	The name of the PVC.
accessModes	The access mode of the PVC.
storage	The capacity claimed by the PVC. The claimed capacity cannot exceed the capacity of the PV.
matchLabels	The labels that are used to select a PV and bind it to the PVC. The labels must be the same as those of the PV.

2. Run the following command to create a PVC:

```
kubectl create -f pvc-static.yaml
```

In the left-side navigation pane, choose **Volumes > Persistent Volume Claims**. On the **Persistent Volume Claims** page, you can view the created PVC.

Step 3: Create an application

Deploy an application named nginx-static and mount the PVC to the application. In this example, an NGINX application is created.

1. Create an *nginx-static.yaml* file and copy the following content into the file:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: web
spec:
 selector:
  matchLabels:
   app: nginx
 serviceName: "nginx"
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
     name: web
    volumeMounts:
    - name: pvc-disk
     mountPath: /data
   volumes:
    - name: pvc-disk
     persistentVolumeClaim:
      claimName: csi-pvc
```

   - `mountPath`: the path where the disk is mounted in the container.
   - `claimName`: the name of the PVC that is mounted to the application.

2. Run the following command to create an application and mount a statically provisioned disk volume by using a PVC:

```
kubectl apply -f nginx-static.yaml
```

In the left-side navigation pane of the details page, choose **Workloads > StatefulSets**. You can view the created application on the **StatefulSets** page.

Verify that the statically provisioned disk can be used to persist data

1. View the pod where the nginx-static application is deployed and files in the disk.

i. Run the following command to query the names of pods that run the nginx-static application:

```
kubectl get pod | grep nginx-static
```

Expected output:

```
nginx-xxxxxx 1/1 Running 0 32s
```

ii. Run the following command to check whether a new disk is mounted to the */data* path:

```
kubectl exec nginx-xxxxxx df | grep data
```

Expected output:

```
/dev/vdf 20511312 45080 20449848 1% /data
```

iii. Run the following command to query files in the */data* path:

```
kubectl exec nginx-xxxxxx ls /data
```

Expected output:

```
lost+found
```

2. Run the following command to create a file named *static* in the */data* path:

```
kubectl exec nginx-xxxxxx touch /data/static
```

3. Run the following command to query files in the */data* path:

```
kubectl exec nginx-xxxxxx ls /data
```

Expected output:

```
staticlost+found
```

4. Run the following command to delete the `nginx-xxxxxx` pod:

```
kubectl delete pod nginx-xxxxxx
```

Expected output:

```
pod "nginx-xxxxxx" deleted
```

5. Verify that the file still exists in the disk after the pod is deleted.

i. Run the following command to query the name of the recreated pod:

```
kubectl get pod
```

Expected output:

```
NAME READY STATUS RESTARTS AGEnginx-xxxxxx 1/1 Running 0 14s
```

ii. Run the following command to query files in the */data* path:

```
kubectl exec nginx-xxxxxx ls /data
```

Expected output:

```
nginx-staticlost+found
```

The *static* file still exists in the disk. This indicates that data is persisted to the statically provisioned disk.

8.4.3. Automatically expand a disk volume

In Kubernetes 1.16, the feature of dynamically expanding disk volumes is in public preview. Container Service for Kubernetes (ACK) allows you to dynamically expand a mounted disk volume by using Container Storage Interface (CSI) in Kubernetes 1.16 and later. This topic describes how to dynamically expand a mounted disk by using CSI.

Context

The expansion includes the expansion of the **disk** size and the expansion of the **file system**. Both expansions can be performed without the need to stop the application where the disk volume is mounted (the disk and file system are still mounted during the expansion). However, to ensure the stability of the file system and application, we recommend that you stop the application, unmount the persistent volume (PV) from the directory, and then expand the PV.

Terms

- Automatic expansion
  You only need to modify the size specified in the persistent volume claim (PVC). The modification is automatically implemented to expand the corresponding PV and disk.
- Manual expansion
  You need to manually expand the PV and run the `resize2fs` command to expand the file system.
- Expansion without stopping the application
  You need to expand the mounted disk and file system without stopping the application.
- Expansion after stopping the application
  You need to expand the mounted disk volume and file system after stopping the application.

Kubernetes 1.16 and later allow you to expand a PV without stopping the pod to which the PV is mounted.

Instruction

- **Limits**
  You can dynamically expand only disks that are smaller than 2,000 GiB.
- **Data backup**
  To avoid data loss caused by errors during the expansion, we recommend that you first create a snapshot for the PV to back up the disk data.
- **Scenarios**
  - Dynamic expansion is applicable to only dynamically provisioned PVs, which are mounted by using PVCs that contain the StorageClassName parameter.
  - ACK does not allow you to expand inline disk volumes. Inline disk volumes are not created by using PVs and PVCs.
  - ACK does not allow you to dynamically expand volumes that are associated with basic disks.
  - Specify **AllowVolumeExpansion: True** for the StorageClass. The AllowVolumeExpansion parameter is automatically set to True for StorageClasses that are created by ACK. For StorageClasses that are manually created, you must manually set the AllowVolumeExpansion parameter to True.

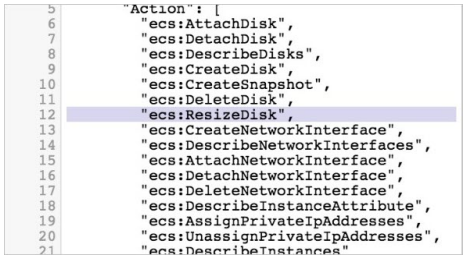
- **Plug-in version**
  Make sure that the FlexVolume or CSI plug-in is upgraded to the latest version.

Grant the ResizeDisk permission to the RAM role of the cluster

Before you dynamically expand a mounted disk without stopping the application, you must grant the ResizeDisk permission to the **RAM role** of the cluster. Perform the following steps based on the cluster type and the volume plug-in that is used:

Dedicated Kubernetes cluster that uses CSI

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane, click **Cluster Information**.
5. Click the **Cluster Resources** tab and click the hyperlink next to **Master RAM Role**.
6. In the RAM console, grant the ResizeDisk permission to the RAM role. For more information, see Modify a custom policy.



Managed or dedicated Kubernetes cluster that uses FlexVolume

Perform the preceding Step 1 to Step 4 and click the hyperlink next to **Master RAM Role**.

Expand a mounted disk volume without stopping the application

1. Connect to a Kubernetes cluster by using kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

   In this example, the pod that you want to manage is in the following state.
   Run the following command to query information about the pod:

   ```
   kubectl get pod
   ```

   Expected output:

   ```
   web-0    1/1   Running  0     42s
   ```

   Run the following command to view the mounting state of the pod:

   ```
   kubectl exec web-0 df /data
   ```

   Expected output:

   ```
   Filesystem    1K-blocks  Used Available Use% Mounted on
   /dev/vdb      20511312 45080  20449848   1% /data
   ```

   Run the following command to query information about the PVC:

   ```
   kubectl get pvc
   ```

Expected output:

```
NAME        STATUS  VOLUME           CAPACITY  ACCESS MODES  STORAGECLASS         AGE
disk-ssd-web-0  Bound   d-wz9hpoifm43yn9zie6gl  20Gi     RWO       alicloud-disk-available  57s
```

Run the following command to query information about the PV:

```
kubectl get pv
```

Expected output:

```
NAME            CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM           STORAGECLASS
REASON  AGE
d-wz9hpoifm43yn9zie6gl  20Gi     RWO       Delete      Bound    default/disk-ssd-web-0  alicloud-disk-av
ailable       65s
```

2. Make sure that the requirements described in the Instruction section are met, and then run the following command to expand the PV:

```
kubectl patch pvc disk-ssd-web-0 -p '{"spec":{"resources":{"requests":{"storage":"30Gi"}}}}'
```

Wait 1 minute and then check whether the PV is expanded.
Run the following command to query information about the PV:

```
kubectl get pv d-wz9hpoifm43yn9zie6gl
```

Expected output:

```
NAME            CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM           STORAGECLASS        R
EASON  AGE
d-wz9hpoifm43yn9zie6gl  30Gi     RWO       Delete      Bound   default/disk-ssd-web-0  alicloud-disk-availa
ble       5m23s
```

Run the following command to query information about the PVC:

```
kubectl get pvc
```

Expected output:

```
NAME        STATUS  VOLUME           CAPACITY  ACCESS MODES  STORAGECLASS         AGE
disk-ssd-web-0  Bound   d-wz9hpoifm43yn9zie6gl  30Gi     RWO       alicloud-disk-available  5m10s
```

Run the following command to query the mounting state of the pod:

```
kubectl exec web-0 df /data
```

Expected output:

```
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/vdb       30832548 45036  30771128   1% /data
```

To expand a mounted disk volume without stopping the application, you need to run only the preceding command.

8.4.4. Manually expand a disk volume

In a Container Service for Kubernetes (ACK) cluster that runs Kubernetes earlier than 1.16, you cannot set the cluster to automatically expand disk volumes. You must manually expand disk volumes. This topic describes how to manually expand a disk volume.

Context

Terms

- Automatic expansion
  You need only to modify the volume size specified in a persistent volume claim (PVC). The modification is automatically applied to expand the corresponding disk volume and the file system.
- Manual expansion
  You must manually expand the disk volume and run the `resize2fs` command to expand the file system.
- Expansion without service interruption
  You can expand the disk volume and file system without the need to stop the application.
- Expansion with service interruption
  You must first stop the application, expand the disk volume and file system, and then start the application again.

To expand a disk volume, perform the following operations:

- Expand the size of the disk: You must perform this operation in the Elastic Compute Service (ECS) console.
- Expand the file system: You must connect to the ECS instance to which the disk is mounted and then perform this operation.
- Modify the volume size specified in the persistent volume (PV) and PVC. This operation is not supported in current versions.

> **Note**
> Disk volumes cannot be automatically expanded in earlier Kubernetes versions due to the following reasons. We recommend that you upgrade Kubernetes to the latest version and then modify the volume size specified in the PV and PVC to automatically expand the disk volume.
>
> - The procedure for modifying the volume size specified in the PV and PVC varies based on the Kubernetes version.
> - The volume size specified in the PV and PVC does not affect the use of the underlying storage device. This means that if the volume size specified in a PV and a PVC is 20 GiB and the sizes of the disk and file system are 30 GiB, the application can use at most 30 GiB of storage.

To ensure stability, Container Service for Kubernetes (ACK) allows you to use the following methods to expand a disk volume:

- Manually expand a disk volume without service interruption: If the I/O throughput of the disk is high when you expand the file system, an I/O error may occur in the file system. You do not need to restart the application if you choose this method.
- Manually expand a disk volume with service interruption: After the application is stopped, the disk I/O operations are stopped. This ensures data security when you expand the file system. Your service will be temporarily interrupted if you choose this method.

Usage notes

- **Limits**
  You can expand only disk volumes that are smaller than 2,000 GiB.
- **Data backup**
  Before you expand a disk volume, you must back up the disk data by creating a snapshot of the disk. This prevents data loss when you expand the disk volume.
- **Scenarios**
  You cannot automatically expand disk volumes in the following scenarios:
  - The Kubernetes version of the cluster is earlier than 1.16.

- The PV is a statically provisioned disk volume.

Examples

A stateful application named web is used in this example to demonstrate how to expand a disk volume by using the preceding two methods. Perform the following operations to query information about the disk:

- Run the following command to query the pods that are provisioned for the web application:

  ```
  kubectl get pod | grep web
  ```

  Expected output:

  ```
  NAME             READY  STATUS  RESTARTS  AGE
  web-0            1/1    Running 0     11h
  web-1            1/1    Running 0     11h
  ```

- Run the following command to query the PVCs that are created for the web application:

  ```
  kubectl get pvc | grep web
  ```

  Expected output:

  ```
  NAME       STATUS  VOLUME     CAPACITY  ACCESS MODES  STORAGECLASS     AGE
  disk-ssd-web-0  Bound  d-0jlhaq***  20Gi    RWO       alicloud-disk-essd  11h
  disk-ssd-web-1  Bound  d-0jl0j5***  20Gi    RWO       alicloud-disk-essd  11h
  ```

- Run the following command to query the PVs that are created for the web application:

  ```
  kubectl get pv | grep web
  ```

  Expected output:

  ```
  NAME     CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM          STORAGECLASS     REASON
  AGE
  d-0jl0j5***  20Gi    RWO      Delete      Bound   default/disk-ssd-web-1  alicloud-disk-essd      11h
  d-0jlhaq***  20Gi    RWO      Delete      Bound   default/disk-ssd-web-0  alicloud-disk-essd      11h
  ```

The output indicates that two disks named `d-0jl0j5***` and `d-0jlhaq***` are used by the web application. Both disks are 20 GiB in size. The disks are mounted to two pods separately.

For more information about how to deploy a stateful application, see Use a StatefulSet to create a stateful application.

Method 1: Expand the disk volume without service interruption

Find the corresponding disk based on the PV information, manually expand the disk, and then connect to the node to which the disk is mounted and expand the file system. The following example demonstrates how to expand both disks to 30 GiB.

Step 1: Expand the disks

1. Log on to the ECS console.
2. In the left-side navigation pane, choose **Storage & Snapshots > Disks**.
3. Find the disks named *d-0jl0j5**** and *d-0jlhaq**** and check the states of the disks. Then, choose **More > Resize Disk** in the **Actions** column for each disk.
4. On the **Resize Disks** page, select **Online Resizing**, and enter the size to which you want to expand the disk in the Size after Resize section. In this example, set the size to 30 GiB.

   > **Note** The specified value cannot be smaller than the current disk size.

5. Confirm the disk expansion fee. Read and select ECS Service Terms, and then click **Confirm**.

   For more information, see Resize disks online for Linux instances.

Step 2: Expand the file systems

After the disks are expanded, you must expand the file systems. Otherwise, the storage that the application can use is still 20 GiB.

> **Notice** This step is intended for unpartitioned disks that are used in Kubernetes. We recommend that you do not use partitioned disks in Kubernetes.
>
> - If an unpartitioned disk is mounted as a PV, you cannot manually create partitions for the disk. Otherwise, the file system may be damaged and data loss may occur.
> - If a partitioned disk is mounted as a PV, you must expand the file system after you can expand the partitioned disk. For more information, see Step 3: View the disk partitions and Step 4: Resize partitions.

1. View the ECS instances to which the disks are mounted.
   i. Log on to the ECS console.
   ii. In the left-side navigation pane, choose **Storage & Snapshots > Disks**.
   iii. Find the disks named *d-0jl0j5**** and *d-0jlhaq****, and click the name of each disk.
   iv. On the **Details** page, click **Attached To** in the **Attaching Information** section.
   v. On the **Instance Details** tab, view **Network Information** about the ECS instance.

   > **Note** You can also view the ECS instances to which the disks are mounted in the ACK console. For more information, see View pods.

2. Connect to the ECS instance to which the disk is mounted and obtain the driver letter of the disk.

   For more information about how to connect to an ECS instance, see Methods used to connect to ECS instances.
   You can use the following methods to obtain the driver letter of the disk.
   - Obtain the driver letter of the disk.
   - Run the following command to query the driver letter of the disk named *d-0jlhaq****:

     ```
     # Query {pv-name}
     mount |grep d-0jlhaq***
     ```

     Expected output:

     ```
     /dev/vdc on /var/lib/kubelet/plugins/kubernetes.io/csi/pv/d-0jlhaq***/globalmount type ext4 (rw,relatime
     )
     /dev/vdc on /var/lib/kubelet/pods/a26d174f-***/volumes/kubernetes.io~csi/d-0jlhaq***/mount type ext4 (
     rw,relatime)
     ```

     The output indicates that the driver letter of the `d-0jlhaq***` disk is */dev/vdc*.

3. Run the following command to expand the file system:

   ```
   resize2fs /dev/vdc
   ```

   > **Note** */dev/vdc* is the driver letter obtained in Step 2.

   Expected output:

```
resize2fs 1.43.5 (04-Aug-2017)
Filesystem at /dev/vdc is mounted on /var/lib/kubelet/plugins/kubernetes.io/csi/pv/d-0jlhaq***/globalmount
; on-line resizing required
old_desc_blocks = 3, new_desc_blocks = 4
The filesystem on /dev/vdc is now 7864320 (4k) blocks long.
```

4. Run the following command to check whether the file system is expanded:

   ```
   lsblk /dev/vdc
   ```

   Expected output:

   ```
   NAME MAJ:MIN  RM SIZE RO TYPE MOUNTPOINT
   vdc  254:32   0  30G  0 disk /var/lib/kubelet/pods/a26d174f-***/volumes/kubernetes.io~csi/d-0jlhaq***/mount
   ```

   The output indicates that the size of the *vdc* file system is expanded to 30 GiB.

Method 2: Expand a disk volume with service interruption

You can stop the application by deleting the StatefulSet or set the value of Replica to 0. Then, you can manually expand the disks and restart the application. The following example demonstrates how to expand both disks to 30 GiB.

Step 1: Delete the pods that are provisioned for the application

1. Run the following command to scale the number of pods to 0:

   ```
   kubectl scale sts web --replicas=0
   ```

   Expected output:

   ```
   statefulset.apps/web scaled
   ```

2. Run the following command to check whether the pods are deleted:

   ```
   kubectl get pod | grep web
   ```

   No output is returned. This indicates that the web application is stopped.

Step 2: Expand the disks

1. Log on to the ECS console.
2. In the left-side navigation pane, choose **Storage & Snapshots > Disks**.
3. Find the disks named *d-0jl0j5**** and *d-0jlhaq**** and check the states of the disks. Then, choose **More > Resize Disk** in the **Actions** column for each disk.
4. On the **Resize Disks** page, select a method to resize the disk, and specify the size to which you want to expand the disk.
   - If the disk is in the **Unattached** state, do not select **Online Resizing** on the **Resize Disks** page. Specify the size to which you want to expand the disk in the Size after Resize section. In this example, set the value to 30 GiB.
   - If the disk is in the **In Use** state, select **Online Resizing** on the **Resize Disks** page, and specify the size to which you want to expand the disk in the Size after Resize section.

   > **Note** The specified value cannot be smaller than the current disk size.

5. Confirm the disk expansion fee. Read and select ECS Service Terms, and then click **Confirm**.

   For more information, see Resize disks online for Linux instances.

Step 3: Expand the file systems

After the disks are expanded, you must expand the file systems. Otherwise, the storage that the application can use is still 20 GiB.

Notice This step is intended for unpartitioned disks that are used in Kubernetes. We recommend that you do not use partitioned disks in Kubernetes.

- If an unpartitioned disk is mounted as a PV, you cannot manually create partitions for the disk. Otherwise, the file system may be damaged and data loss may occur.
- If a partitioned disk is mounted as a PV, you must expand the file system after you can expand the partitioned disk. For more information, see Step 3: View the disk partitions and Step 4: Resize partitions.

1. (Optional)Mount the disks to an ECS instance.

   Note You must mount the disks to an ECS instance before you can expand the file systems.

   i. Log on to the ECS console.
   ii. In the left-side navigation pane, choose **Storage & Snapshots > Disks**.
   iii. Find the disk that is in the **Unattached** state and choose **More > Attach** in the **Actions** column.
   iv. In the **Attach Disk** dialog box, select an ECS instance from the drop-down list, and configure the settings that correspond to releasing disks.

Parameter	Description
Target Instance	Select the ECS instance to which you want to mount the disk.
Release Disk with Instance	If you select this option, the disk is automatically released when the ECS instance to which it is mounted is released. If you do not select this option, the disk is retained when the ECS instance to which it is mounted is released. **Note** If the disk that you want to mount is a system disk that you unmounted from another ECS instance, the ECS instance specified by **Release Disk with Instance** refers to the ECS instance from which you unmounted the disk.
Delete Automatic Snapshots While Releasing Disk	If you select this option, snapshots that are automatically created for the disk are released together with disk. To retain the snapshots, do not select this option.

   v. Click **Attach**.

   If the status of the disk becomes **In Use**, the disk is attached.

2. Connect to the ECS instance to which the disk is mounted and obtain the driver letter of the disk.

   For more information about how to connect to an ECS instance, see Methods used to connect to ECS instances.

   - Run the following command to obtain the driver letter of the disk:

```
for device in `ls /sys/block | grep vd`; do
 cat /sys/block/$device/serial | grep 0jlhaq*** && echo $device;
done
```

> **Note** The ID of the expanded disk is `d-0jlhaq***`. `0jlhaq***` is the string that follows `d-`.

- (Optional)If you cannot obtain the driver letter of the disk by running the preceding command, perform the following operations:
  a. Unmount the disk and run the `ls /dev/vd*` command to query the list of disks.
  b. Mount the disk and run the `ls /dev/vd*` command to query the list of disks.
  c. Compare the lists that are returned. The disk that appears only in the second list is the one that you mounted.

3. Run the following command to expand the file system:

```
resize2fs /dev/vdb
```

> **Note** */dev/vdc* is the driver letter of the disk obtained in the Step 2.

Expected output:

```
resize2fs 1.43.5 (04-Aug-2017)
Resizing the filesystem on /dev/vdb to 7864320 (4k) blocks.
The filesystem on /dev/vdb is now 7864320 (4k) blocks long.
```

4. Check whether the file system is expanded.
   i. Run the following command to create a temporary folder named */mnt/disk/* and mount the disk to the folder:

```
mkdir /mnt/disk
mount /dev/vdb /mnt/disk/
```

   ii. Run the following command to query the size of the specified file system:

```
df /mnt/disk/
```

   Expected output:

```
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/vdb       30832548 45036  30771128   1% /mnt/disk
```

   The output indicates that the */dev/vdb* folder can use at most 30 GiB of storage. This indicates that the file system is expanded.

   iii. Run the following command to unmount the disk from the temporary folder:

```
umount /mnt/disk
```

Step 4: Restart the application

1. Run the following command to scale the number of pods to 2:

```
kubectl scale sts web --replicas=2
```

Expected output:

```
statefulset.apps/web scaled
```

2. Run the following command to check whether the pods are deleted:

```
kubectl get pod | grep web
```

Expected output:

```
NAME    READY  STATUS  RESTARTS  AGE
web-0   1/1    Running 0     74s
web-1   1/1    Running 0     42s
```

3. Run the following command to query the size of the specified file system:

```
kubectl exec web-0 df /data
```

Expected output:

```
Filesystem    1K-blocks  Used Available Use% Mounted on
/dev/vdb      30832548 45036  30771128   1% /data
```

The output indicates that the size of the */dev/vdb* file system is expanded to 30 GiB.

FAQ

Issue:

What can I do if the following message appears after I run the `resize2fs` command?

```
resize of device /dev/xxx failed: exit status 1 resize2fs output: resize2fs xxx(version)
Please run `e2fsck -f /dev/xxx` first
```

Cause:

The file systems are not consistent, which causes I/O errors.

Solution:

Run the `e2fsck -f /dev/xxx` command and then expand the file systems.

Related information

- Automatically expand a disk volume

8.4.5. Use volume snapshots created from disks

Container Service for Kubernetes (ACK) allows you to create volume snapshots from disks and use the volume snapshots to restore application data. This topic describes the basic terms related to volume snapshots and how to use volume snapshots in ACK clusters. This topic also describes how to dynamically and statically create volume snapshots.

Prerequisites

- ACK clusters that use Kubernetes V1.18 and later support volume snapshots. To use volume snapshots that are created from disks, make sure that Kubernetes V1.18 or later is used in your ACK cluster. For more information, see 创建Kubernetes托管版集群.
- Log on the Elastic Compute Service (ECS) console and select the region where your cluster is deployed. Make sure that the volume snapshot feature is enabled. For more information, see Activate ECS Snapshot.

Context

When you deploy stateful applications in an ACK cluster, disks are commonly used to store application data. You can create snapshots from disks and use these snapshots to restore application data. ACK also allows you to use snapshots to back up and restore data. You can back up and restore data in your ACK cluster in the following ways:

- Create backups (volume snapshots) from disks by using VolumeSnapshots.
- Restore data by setting dataSource to the related persistent volume claim (PVC).

Instruction

To enable volume snapshots, you can use CustomResourceDefinitions (CRDs) in ACK to define the following resources:

Resource name	Description
VolumeSnapshotContent	A snapshot that is created from a volume in the ACK cluster. It is created and managed by administrators. A VolumeSnapshotContent does not belong to any namespace. A VolumeSnapshotContent is a cluster resource similar to a persistent volume (PV).
VolumeSnapshot	A request for a volume snapshot. It is created and managed by users. A VolumeSnapshot belongs to a specific namespace. A VolumeSnapshot is a cluster resource similar to a PVC.
VolumeSnapshotClass	Specifies the attributes of a VolumeSnapshot, such as the parameters and controllers that are used to create snapshots. A VolumeSnapshotClass is a cluster resource similar to a StorageClass.

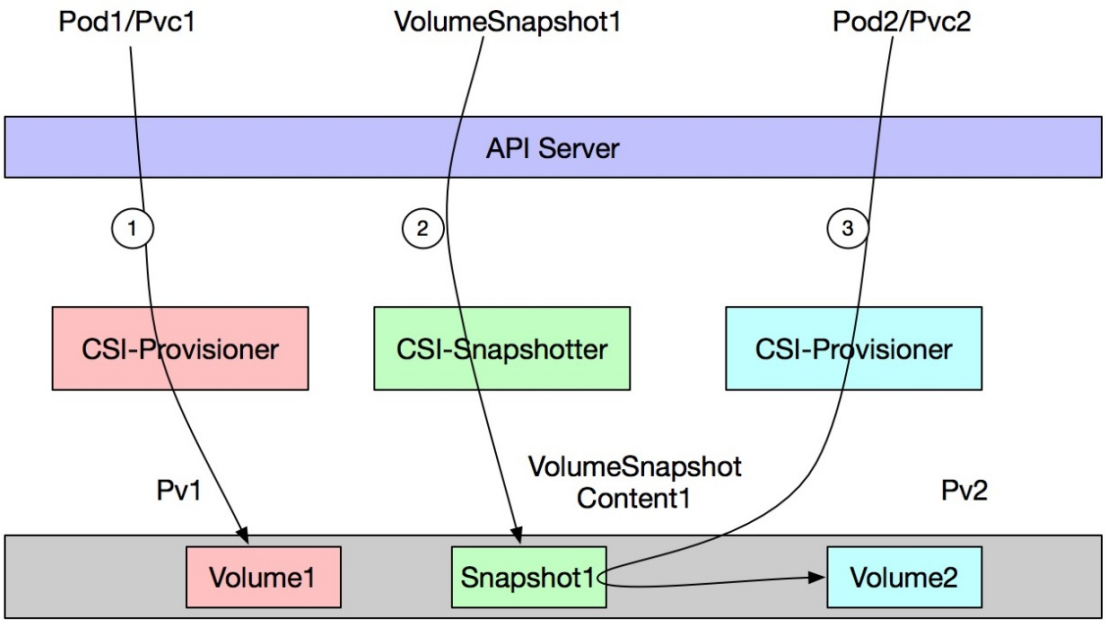
To use volume snapshots, you must associate these resources with each other based on the following rules:

- Before you can use volume snapshots, associate a VolumeSnapshot with a VolumeSnapshotContent in the same way that you associate a PV with a PVC.
- After you specify a valid value in the VolumeSnapshotClassName field for a VolumeSnapshot, the ACK cluster automatically creates a VolumeSnapshotContent for the VolumeSnapshot. If you specify an invalid value or do not specify a value in the VolumeSnapshotClassName field, the VolumeSnapshotContent cannot be automatically created. In this case, you must manually create a VolumeSnapshotContent and associate the VolumeSnapshotContent with the VolumeSnapshot.
- Each VolumeSnapshotContent can be associated with only one VolumeSnapshot.

Notice When you delete a VolumeSnapshotContent, the associated snapshot is also deleted.

Dynamically create a volume snapshot

The following figure shows the procedure to dynamically create a volume snapshot from the disk that is used in an ACK cluster.



The following table describes the steps that are performed in the procedure.

Step	Description
①	Create an application and create a disk volume to store application data.
②	Create a VolumeSnapshot. Then, the ACK cluster automatically creates a VolumeSnapshotContent and a volume snapshot.
3	Create another application, create a PVC for the application, and then specify the volume snapshot that is created in Step 2 as the source of the PVC.

The preceding steps are performed to enable the following features:

- Backup: **Snapshot 1** is created to back up data in **Volume 1**.
- Restoration: **Snapshot 1** is used to restore data in **Volume 1** to **Volume 2**.

The following example shows how to create an NGINX application and restore the data of the application by using a volume snapshot. The following procedure demonstrates how to use a volume snapshot to restore the data of an application.

1. Create a VolumeSnapshotClass.

i. Create a *volumesnapshotclass.yaml* file and copy the following content to the file:

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: default-snapclass
driver: diskplugin.csi.alibabacloud.com
parameters:
  snapshotType: "xxx"
  instantAccessRetentionDays: "1"
deletionPolicy: Delete
```

Parameter	Description
deletionPolicy	■ If the value is set to *Delete*, the corresponding VolumeSnapshotContent and snapshot are deleted when you delete a VolumeSnapshot. ■ If the value is set to *Retain*, the corresponding VolumeSnapshotContent and snapshot are retained when you delete a VolumeSnapshot.
parameters.snapshotType	If the value is set to *InstantAccess*, the instant access mode is enabled to dynamically create snapshots. This feature is applicable to only enhanced SSDs. For more information, see Enable or disable the instant access feature. The default value is *normal*, which indicates that a normal snapshot is created.
parameters.instantAccessRetentionDays	You must set this parameter if parameters.snapshotType is set to InstantAccess. This parameter specifies the retention period of snapshot that is created in instant access mode. After the feature is disabled, the snapshot that is created in instant access mode is converted to a normal snapshot.

ii. Run the following command to create a VolumeSnapshotClass:

```
kubectl apply -f volumesnapshotclass.yaml
```

2. Create an application and write data to the application.

i. Create an *nginx.yaml* file and copy the following content to the file:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: web
spec:
 selector:
  matchLabels:
   app: nginx
 serviceName: "nginx"
 replicas: 1
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    volumeMounts:
    - name: disk-ssd
     mountPath: /data
 volumeClaimTemplates:
 - metadata:
   name: disk-ssd
  spec:
   accessModes: [ "ReadWriteOnce" ]
   storageClassName: "alicloud-disk-ssd"
   resources:
    requests:
     storage: 20Gi
```

ii. Run the following command to create an NGINX application:

```
kubectl apply -f nginx.yaml
```

iii. Run the following command to write data to pod web-0:

```
kubectl exec -it web-0 touch /data/test
kubectl exec -it web-0 ls /data
```

The following output is returned:

```
lost+found test
```

3. Create a VolumeSnapshot.

i. Create a *snapshot.yaml* file and copy the following content to the file:

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
 name: new-snapshot-demo
spec:
 volumeSnapshotClassName: default-snapclass
 source:
  persistentVolumeClaimName: disk-ssd-web-0
```

ii. Run the following command to create a VolumeSnapshot:

```
kubectl apply -f snapshot.yaml
```

iii. Run the following command to check whether the VolumeSnapshot and VolumeSnapshotContent are created. You can also log on to the ECS console to view the snapshot.

Run the following command to view the VolumeSnapshot:

```
kubectl get volumesnapshots.snapshot.storage.k8s.io
```

The following output is returned:

```
NAME                AGE
new-snapshot-demo   36m
```

Run the following command to view the VolumeSnapshotContent:

```
kubectl get VolumeSnapshotContent
```

The following output is returned:

```
NAME                      AGE
snapshotcontent-222222    36m
```

4. Restore data.

i. Create an *nginx-restore* file and copy the following content to the file:

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 labels:
  app: nginx
spec:
 ports:
 - port: 80
  name: web
 clusterIP: None
 selector:
  app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: web-restore
spec:
 selector:
  matchLabels:
   app: nginx
 serviceName: "nginx"
 replicas: 1
 template:
  metadata:
   labels:
    app: nginx
  spec:
   hostNetwork: true
   containers:
   - name: nginx
    image: nginx
    command: ["sh", "-c"]
    args: ["sleep 10000"]
    volumeMounts:
    - name: disk-ssd
     mountPath: /data
 volumeClaimTemplates:
 - metadata:
   name: disk-ssd
  spec:
   accessModes: [ "ReadWriteOnce" ]
   storageClassName: alicloud-disk-ssd
   resources:
    requests:
     storage: 20Gi
   dataSource:
    name: new-snapshot-demo
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

> **Note** In the dataSource field of the volumeClaimTemplates section, set kind to VolumeSnapshot and set name to new-snapshot-demo. The new-snapshot-demo snapshot is created in Step .

ii. Run the following command to restore the application data:

```
kubectl apply -f nginx-restore.yaml
```

5. Run the following command to view the application data in pod web-restore-0:

```
kubectl exec -it web-restore-0 ls /data
```

The following output is returned:

```
lost+found test
```

The output shows that the same application data is returned. This indicates that the application data is restored.

Statically create a volume snapshot (import an existing snapshot from an ECS instance)

The following procedure demonstrates how to import an existing snapshot from an ECS instance to an ACK cluster.

1. Use the following YAML template to create a VolumeSnapshotContent:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
 name: new-snapshot-content-test
spec:
 deletionPolicy: Retain
 driver: diskplugin.csi.alibabacloud.com
 source:
  snapshotHandle: <your-snapshotid>
 volumeSnapshotRef:
  name: new-snapshot-demo
  namespace: default
```

Parameter	Description
snapshotHandle	Enter the snapshot ID that is generated on the ECS instance page.
volumeSnapshotRef	Enter the following information for the VolumeSnapshot that you want to create: ○ name: the name of the VolumeSnapshot. ○ namespace: the namespace to which the VolumeSnapshot belongs.

2. Use the following YAML template to create a VolumeSnapshot:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
 name: new-snapshot-demo
spec:
 source:
  volumeSnapshotContentName: new-snapshot-content-test
```

Parameter	Description
metadata.name	The name of the VolumeSnapshot. The name must be the same as the VolumeSnapshot name that is specified in the preceding VolumeSnapshotContent.
spec.source.volumeSnapshotContentName	The name of the VolumeSnapshotContent that is associated with the VolumeSnapshot. The name must be the same as the name of the VolumeSnapshotContent that is created in the preceding step.

3. Restore data.
   i. Create an *nginx-restore* file and copy the following content to the file:

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 labels:
  app: nginx
spec:
 ports:
 - port: 80
  name: web
 clusterIP: None
 selector:
  app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: web-restore
spec:
 selector:
  matchLabels:
   app: nginx
 serviceName: "nginx"
 replicas: 1
 template:
  metadata:
   labels:
    app: nginx
  spec:
   hostNetwork: true
   containers:
   - name: nginx
    image: nginx
    command: ["sh", "-c"]
    args: ["sleep 10000"]
    volumeMounts:
    - name: disk-ssd
     mountPath: /data
 volumeClaimTemplates:
 - metadata:
   name: disk-ssd
  spec:
   accessModes: [ "ReadWriteOnce" ]
   storageClassName: alicloud-disk-ssd
   resources:
    requests:
     storage: 20Gi
   dataSource:
    name: new-snapshot-demo
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

> **Note** In the dataSource field of the volumeClaimTemplates section, set kind to VolumeSnapshot and set name to new-snapshot-demo. The new-snapshot-demo snapshot is created in Step 2.

ii. Run the following command to restore the application data:

```
kubectl apply -f nginx-restore.yaml
```

4. Run the following command to view the application data in pod web-restore-0:

```
kubectl exec -it web-restore-0 ls /data
```

The following output is returned:

```
lost+found test
```

The output shows that the same application data is returned. This indicates that the application data is restored.

8.5. NAS volumes

8.5.1. Overview

Container Service for Kubernetes (ACK) allows you to mount Apsara File Storage NAS (NAS) file systems as persistent volumes (PVs) in ACK clusters. This topic describes the limits of NAS volumes and provides usage notes for NAS volumes.
NAS file systems can be mounted to an ACK cluster by using the Container Storage Interface (CSI) plug-in in two forms:

- Mount as statically provisioned PVs
- Mount as dynamically provisioned PVs

Prerequisites

A NAS file system is created and a mount target is added to the file system. To create a NAS file system and add a mount target, log on to the NAS console. The mount target of the NAS file system and your cluster are deployed in the same virtual private cloud (VPC).

The mount target is in the following format: `055f84ad83-ixxxx.cn-hangzhou.nas.aliyuncs.com`.

Usage notes

- Apsara File Storage NAS is a shared storage service. A persistent volume claim (PVC) that is used to mount a NAS file system can be shared among pods.
- Do not delete the mount target before you unmount the NAS file system. Otherwise, the operating system hang may occur.
- After a mount target is created, wait until the mount target is ready for use.
- We recommend that you use NFSv3.
- We also recommend that you upgrade the CSI plug-in to the latest version before you mount NAS file systems as PVs.
- General-purpose and Extreme NAS file systems have different limits on mounting scenarios, the number of file systems, and file sharing protocols. For more information, see Limits of Apsara File Storage NAS.

8.5.2. Use a statically provisioned NAS volume

Apsara File Storage NAS (NAS) is a distributed file system that supports shared access. NAS volumes are scalable, high-performance, and highly reliable. This topic describes how to use a statically provisioned NAS volume, and how to enable persistent storage and shared storage by using a statically provisioned NAS volume.

Prerequisites

- A Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群.
- A NAS file system is created. For more information, see Create a NAS file system.
  To encrypt the data in a NAS volume, configure the encryption settings when you create the corresponding NAS file system.
- A mount target is created for the NAS file system. For more information, see Manage mount targets.
  The mount target and the cluster node to which you want to mount the NAS file system must belong to the same virtual private cloud (VPC).
- Your machine is connected to the cluster by using kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

Scenarios:

- You want to run applications that require high disk I/O.
- You need a storage service that provides higher read and write performance than Object Storage Service (OSS).
- You want to share files across hosts. For example, you want to use a NAS file system as a file server.

Precaution

- To mount an Extreme NAS file system, set the `path` parameter of the NAS volume to a subdirectory of */share*. For example, you can specify the */share/path1* subdirectory when you mount an Extreme NAS file system to a pod.
- If a NAS file system is mounted to multiple pods, data in the file system is shared by these pods. In this case, the application must be capable of automatically synchronizing data across pods.

> **Note** You cannot grant permissions to access the / directory (root directory) of the NAS file system. The user account and user group to which the directory belongs cannot be modified.

Use a NAS file system as a statically provisioned volume in the ACK console

Step 1: Create a PV

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.
5. In the upper-right corner of the **Persistent Volumes** page, click **Create**.
6. In the **Create PV** dialog box, set the following parameters.

Parameter	Description
PV Type	You can select Cloud Disk, NAS, or OSS. In this example, NAS is selected.

Parameter	Description
Volume Name	The name of the persistent volume (PV) that you want to create. The name must be unique in the cluster. In this example, pv-nas is used.
Volume Plug-in	You can select Flexvolume or CSI. In this example, CSI is selected.
Capacity	The capacity of the PV. A NAS file system provides unlimited capacity. This parameter does not limit the usage of the NAS file system but defines the capacity of the PV.
Access Mode	You can select ReadWriteMany or ReadWriteOnce. Default value: ReadWriteMany.
Mount Target Domain Name	You can **select a mount target** or enter a **custom** mount target.
Show Advanced Options	○ **Subdirectory**: the subdirectory of the NAS file system that you want to mount. The subdirectory must start with a forward slash (/). After you set this parameter, the PV is mounted to the subdirectory. ■ If the specified subdirectory does not exist, the system automatically creates the subdirectory in the NAS file system. ■ If you do not set this parameter, the PV is mounted to the root directory of the NAS file system. ■ If you want to mount an Extreme NAS file system, the subdirectory must be under the */share* directory. ○ **Version**: the version of the PV.
Label	Add labels to the PV.

7. Click **Create**.

Step 2: Create a PVC

1. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volume Claims**.
2. In the upper-right corner of the **Persistent Volume Claims** page, click **Create**.
3. In the **Create PVC** dialog box, set the following parameters.

Parameter	Description
PVC Type	You can select Cloud Disk, NAS, or OSS. In this example, NAS is selected.
Name	The name of the persistent volume claim (PVC). The name must be unique in the cluster.
Allocation Mode	In this example, Existing Volumes is selected. **Note** If no PV is created, you can set **Allocation Mode** to **Create Volume**, and set other parameters to create a PV. For more information, see Create a PV.
Existing Volumes	Click **Select PV**. Find the PV that you want to use and click **Select** in the Actions column.

Parameter	Description
Capacity	The capacity claimed by the PVC. **Note** The capacity claimed by the PVC cannot exceed the capacity of the PV that is bound to the PVC.

4. Click **Create**.

   After the PVC is created, you can view the PVC in the PVC list. The PVC is bound to the corresponding PV.

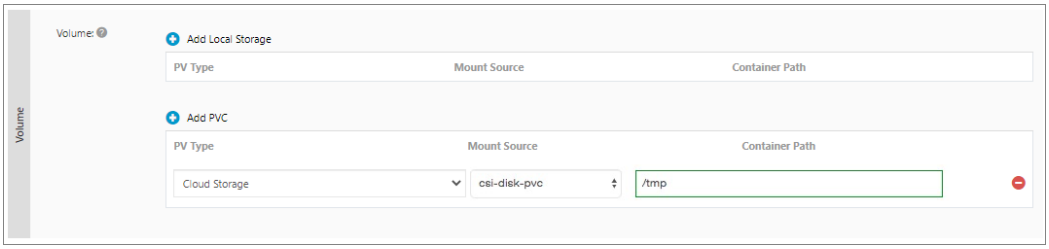
Step 3: Create an application

1. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
2. In the upper-right corner of the **Deployments** page, click **Create from Image**.
3. Set the parameters that are used to create an application.

   This example shows how to set volume parameters in the application configurations. For more information about other parameters, see Create a stateless application by using a Deployment. You can add local volumes and cloud volumes.

   - **Add Local Storage**: You can select HostPath, ConfigMap, Secret, or EmptyDir from the PV Type drop-down list. Then, set the Mount Source and Container Path parameters to mount the volume to a container path. For more information, see Volumes.
   - **Add PVC**: You can add cloud volumes.

   In this example, csi-nas-pvc is specified as the mount source and mounted to the */tmp* path in the container.



4. Set other parameters and click **Create**.

   After the application is created, you can use the NAS volume to store application data.

Use a statically provisioned NAS volume by using kubectl

1. Run the following command to create a statically provisioned PV:

   ```
   kubectl create -f pv-nas.yaml
   ```

   The following YAML template provides an example on how to create a statically provisioned PV:

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: pv-nas
 labels:
  alicloud-pvname: pv-nas
spec:
 capacity:
  storage: 5Gi
 accessModes:
  - ReadWriteMany
 csi:
  driver: nasplugin.csi.alibabacloud.com
  volumeHandle: pv-nas
  volumeAttributes:
   server: "2564f49129-ysu87.cn-shenzhen.nas.aliyuncs.com"
   path: "/csi"
 mountOptions:
 - nolock,tcp,noresvport
 - vers=3
```

Parameter	Description
name	The name of the PV.
labels	The labels that are added to the PV.
storage	The capacity of the NAS volume.
accessModes	The access mode of the PV.
driver	The type of driver. In this example, the parameter is set to `nasplugin.csi.alibabacloud.com`. This indicates that the NAS Container Storage Interface (CSI) plug-in provided by Alibaba Cloud is used.
volumeHandle	The unique identifier of the PV. If multiple PVs are used, the identifier of each PV must be unique.
server	The mount target of the NAS file system.
path	The subdirectory of the NAS file system that is mounted. If you want to mount an Extreme NAS file system, the subdirectory must be under the */share* directory.
vers	The version of the Network File System (NFS) protocol. We recommend that you use NFSv3. Extreme NAS file systems support only NFSv3.

2. Run the following command to create a PVC used for static provisioning:

   When you create a PVC of the NAS type, set the selector parameter to configure how to select a PV and bind it to the PVC.

```
kubectl create -f pvc-nas.yaml
```

The following YAML template provides an example on how to create a PVC used for static provisioning:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: pvc-nas
spec:
 accessModes:
  - ReadWriteMany
 resources:
  requests:
   storage: 5Gi
 selector:
  matchLabels:
   alicloud-pvname: pv-nas
```

Parameter	Description
name	The name of the PVC.
accessModes	The access mode of the PVC.
storage	The capacity claimed by the PVC. The claimed capacity cannot exceed the capacity of the PV bound to the PVC.
mathLabels	The labels are used to select a PV and bind it to the PVC.

3. Run the following command to create an application named **nas-static** and mount the created PVC to the application:

```
kubectl create -f nas.yaml
```

The following YAML tempalte provides an example of the *nas.yaml* file that is used to create the **nas-static** application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nas-static
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
     - name: pvc-nas
      mountPath: "/data"
   volumes:
    - name: pvc-nas
     persistentVolumeClaim:
      claimName: pvc-nas
```

Parameter	Description
mountPath	The path of the container where the NAS volume is mounted.
claimName	The name of the PVC mounted to the application.

4. Run the following command to query the pods that run the application:

```
kubectl get pod
```

Expected output:

```
NAME                READY  STATUS   RESTARTS  AGE
deployment-nas-1-5b5cdb85f6-n****  1/1   Running  0     32s
deployment-nas-2-c5bb4746c-4****   1/1   Running  0     32s
```

Verify that the mounted NAS file system can be used to persist data

1. Query the pods that run the application and the files in the mounted NAS file system.

i. Run the following command to query the pods that run the application:

```
kubectl get pod
```

Expected output:

```
NAME                     READY  STATUS   RESTARTS  AGE
deployment-nas-1-5b5cdb85f6-n****  1/1    Running  0      32s
deployment-nas-2-c5bb4746c-4****   1/1    Running  0      32s
```

ii. Run the following command to query files in the */data* path of a pod. The pod `deployment-nas-1-5b5cdb85f6-n****` is used as an example:

```
kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
```

No output is returned. This indicates that no file exists in the */data* path.

2. Run the following command to create a file named *nas* in the */data* path of the pod `deployment-nas-1-5b5cdb85f6-n****` :

```
kubectl exec deployment-nas-1-5b5cdb85f6-n**** touch /data/nas
```

3. Run the following command to query files in the */data* path of the pod `deployment-nas-1-5b5cdb85f6-n****` :

```
kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
```

Expected output:

```
nas
```

4. Run the following command to delete the pod:

```
kubectl delete pod deployment-nas-1-5b5cdb85f6-n****
```

5. Open another kubectl CLI and run the following command to query how the pod is deleted and recreated:

```
kubectl get pod -w -l app=nginx
```

6. Verify that the file still exists after the pod is deleted.

i. Run the following command to query the name of the recreated pod:

```
kubectl get pod
```

Expected output:

```
NAME                     READY  STATUS   RESTARTS  AGE
deployment-nas-1-5b5cdb85f6-n****  1/1    Running  0      32s
deployment-nas-2-c5bb4746c-4****   1/1    Running  0      32s
```

ii. Run the following command to query files in the */data* path of the pod `deployment-nas-1-5b5cdb85f6-n****` .

```
kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
```

Expected output:

```
nas
```

The *nas* file still exists in the /data path. This indicates that data is persisted to the NAS file system.

Verify that data in the mounted NAS file system can be shared

1. Query the pods that run the application and the files in the mounted NAS file system.
   i. Run the following command to query the pods that run the application:

   ```
   kubectl get pod
   ```

   Expected output:

   ```
   NAME                READY  STATUS   RESTARTS  AGE
   deployment-nas-1-5b5cdb85f6-n****  1/1   Running  0     32s
   deployment-nas-2-c5bb4746c-4****   1/1   Running  0     32s
   ```

   ii. Run the following command to query files in the */data* path of each pod:

   ```
   kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
   kubectl exec deployment-nas-2-c5bb4746c-4**** ls /data
   ```

2. Run the following command to create a *nas* file in the */data* path of a pod:

   ```
   kubectl exec deployment-nas-1-5b5cdb85f6-n**** touch /data/nas
   ```

3. Run the following command to query files in the */data* path of each pod:
   i. Run the following command to query files in the */data* path of the pod `deployment-nas-1-5b5cdb85f6-n****` :a

   ```
   kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
   ```

   Expected output:

   ```
   nas
   ```

   ii. Run the following command to query files in the */data* path of the pod `deployment-nas-2-c5bb4746c-4****` :

   ```
   kubectl exec deployment-nas-2-c5bb4746c-4**** ls /data
   ```

   Expected output:

   ```
   nas
   ```

   When you create a file in the */data* path of one pod, you can also find the file in the */data* path of the other pod. This indicates that data in the NAS file system is shared by the two pods.

8.5.3. Use a dynamically provisioned NAS volume

You can use the Container Storage Interface (CSI) driver to mount a dynamically provisioned Apsara File Storage NAS (NAS) volume to a Container Service for Kubernetes (ACK) cluster in subpath and filesystem modes. This topic describes how to use a dynamically provisioned NAS volume and how to enable persistent storage and shared storage by using dynamically provisioned NAS volumes.

Prerequisites

- A Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群.
- A NAS file system is created. For more information, see Create a NAS file system.
  If you want to encrypt data in a NAS volume, configure the encryption settings when you create the NAS file system.
- A mount target is created for the NAS file system. For more information, see Manage mount targets.

The mount target and the cluster node to which you want to mount the NAS file system must belong to the same virtual private cloud (VPC).

Scenarios:

- You want to run applications that require high disk I/O.
- You need a storage service that provide higher read and write performance than Object Storage Service (OSS).
- You want to share files across hosts. For example, you want to use a NAS file system as a file server.

Precaution

- To mount an Extreme NAS file system, set the `path` parameter of the NAS volume to a subdirectory of */share*. For example, a value of `xxxxxxx.cn-hangzhou.nas.alivuncs.com:/share/subpath` indicates that the mounted subdirectory of the NAS file system is `/share/subpath`.
- If a NAS file system is mounted to multiple pods, data in the file system is shared by these pods. In this case, the application must be capable of automatically synchronizing data across pods.

> **Note** You cannot grant permissions to access the / directory (root directory) of the NAS file system. The user account and user group to which the directory belongs cannot be modified.

Mount a dynamically provisioned NAS volume in subpath mode

The subpath mode is applicable to scenarios where you need to share a NAS volume among different applications or pods. You can also use this mode to mount different subdirectories of the same NAS file system to different pods.

To mount a dynamically provisioned NAS volume in subpath mode, you must manually create a NAS file system and a mount target.

1. Create a NAS file system and a mount target.
   i. Log on to the NAS console.
   ii. Create a NAS file system. For more information, see Create a NAS file system.
   iii. Create a mount target. For more information, see Manage mount targets.
2. Create a StorageClass.

i. Create an *alicloud-nas-subpath.yaml* file and copy the following content into the file:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: alicloud-nas-subpath
mountOptions:
- nolock,tcp,noresvport
- vers=3
parameters:
 volumeAs: subpath
 server: "xxxxxxx.cn-hangzhou.nas.aliyuncs.com:/k8s/"
provisioner: nasplugin.csi.alibabacloud.com
reclaimPolicy: Retain
```

Parameter	Description
mountOptions	Set the options parameter and specify the Network File System (NFS) version in the mountOptions field.
volumeAs	You can select subpath or filesystem. subpath indicates that a subdirectory is mounted to the cluster. filesystem indicates that a file system is mounted to the cluster.
server	When you mount a subdirectory of the NAS file system as a persistent volume (PV), this parameter specifies the mount target of the NAS file system.
reclaimPolicy	The policy for reclaiming the PV.
archiveOnDelete	This parameter specifies the reclaim policy of the backend storage when reclaimPolicy is set to Delete. NAS is a shared storage service. You must set both reclaimPolicy and archiveOnDelete to ensure data security. The default value is true. This value indicates that the subdirectory or files are not deleted when the PV is deleted. Instead, the subdirectory or files are renamed in the format of `archived-{pvName}.{timestamp}`. If the value is set to false, it indicates that the backend storage resource is deleted when the PV is deleted.

ii. Run the following command to create a StorageClass:

```
kubectl create -f alicloud-nas-subpath.yaml
```

3. Run the following command to create a persistent volume claim (PVC).

i. Create a *nas.yaml* file and copy the following content into the file:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: nas-csi-pvc
 spec:
  accessModes:
   - ReadWriteMany
  storageClassName: alicloud-nas-subpath
  resources:
   requests:
    storage: 20Gi
```

Parameter	Description
name	The name of the PVC.
accessModes	The access mode of the PVC.
storageClassName	The name of the StorageClass, which is used to associate the PVC with the StorageClass.
storage	The storage that is requested by the application.

ii. Run the following command to create a PVC:

```
kubectl create -f pvc.yaml
```

4. Run the following command to create applications.

   Deploy two applications named **nginx-1** and **nginx-2** to share the same subdirectory of the NAS file system.

i. Create an *nginx-1.yml* file and copy the following content into the file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: deployment-nas-1
 labels:
  app: nginx-1
spec:
 selector:
  matchLabels:
   app: nginx-1
 template:
  metadata:
   labels:
    app: nginx-1
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    volumeMounts:
     - name: nas-pvc
      mountPath: "/data"
   volumes:
    - name: nas-pvc
     persistentVolumeClaim:
      claimName: nas-csi-pvc
```

- `mountPath`: the mount path in the container where the NAS volume is mounted.
- `claimName`: the name of the PVC that is mounted to the application. In this case, the value is set to **nas-csi-pvc**.

ii. Create an *nginx-2.yml* file and copy the following content into the file.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: deployment-nas-2
 labels:
  app: nginx-2
spec:
 selector:
  matchLabels:
   app: nginx-2
 template:
  metadata:
   labels:
    app: nginx-2
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    volumeMounts:
     - name: nas-pvc
      mountPath: "/data"
   volumes:
    - name: nas-pvc
     persistentVolumeClaim:
      claimName: nas-csi-pvc
```

- `mountPath`: the mount path in the container where the NAS volume is mounted. In this example, the value is set to */data*.
- `claimName`: Enter the name of the PVC mounted to the **nginx-1** application. In this example, the value is set to **nas-csi-pvc**.

iii. Run the following command to deploy the nginx-1 and nginx-2 applications:

```
kubectl create -f nginx-1.yaml -f nginx-2.yaml
```

5. Run the following command to query the pods:

```
kubectl get pod
```

Expected output:

```
NAME                READY  STATUS   RESTARTS  AGE
deployment-nas-1-5b5cdb85f6-n****  1/1   Running  0     32s
deployment-nas-2-c5bb4746c-4****   1/1   Running  0     32s
```

> **Note** The subdirectory `xxxxxxx.cn-hangzhou.nas.aliyuncs.com:/share/nas-79438493-f3e0-11e9-bbe5-00163e09****` of the NAS volume is mounted to the */data* directory of pods `deployment-nas-1-5b5cdb85f6-n****` and `deployment-nas-2-c5bb4746c-4****`. Where:
>
> - `/share`: the subdirectory is mounted in subpath mode as specified in the StorageClass configurations.
> - `nas-79438493-f3e0-11e9-bbe5-00163e09****`: the name of the PV.
>
> To mount different subdirectories of a NAS file system to different pods, you must create a separate PVC for each pod. In this case, you need to create **pvc-1** for **nginx-1** and **pvc-2** for **nginx-2**.

Mount a dynamically provisioned NAS volume in filesystem mode

> **Notice** By default, if you delete a PV that is mounted in filesystem mode, the system retains the related NAS file system and the mount target. To delete the NAS file system and the mount target together with the PV, set reclaimPolicy to Delete and set deleteVolume to true in the StorageClass configurations.

The filesystem mode is applicable to scenarios where you want to dynamically create and delete NAS file systems and mount targets.

When you mount a NAS volume in filesystem mode, you can create only one NAS file system and one mount target for each pod. You cannot share a NAS volume among multiple pods. The following procedure demonstrates how to mount a dynamically provisioned NAS volume in filesystem mode.

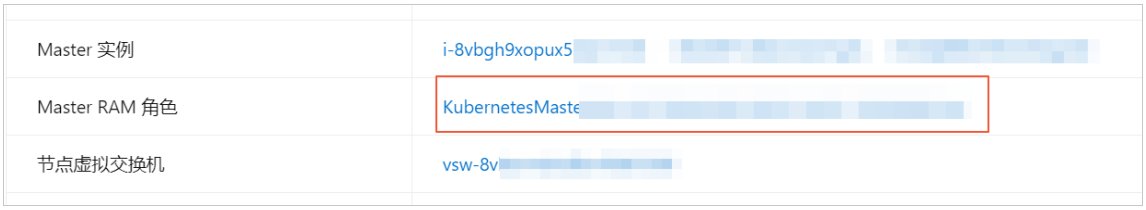
1. Configure a Resource Access Management (RAM) permission policy and attach it to a RAM role.

   The filesystem mode allows you to dynamically create and delete NAS file systems and mount targets. To enable this feature, you must grant the required permissions to csi-nasprovisioner. The following code block shows a permission policy that contains the required permissions:

   ```
   {
     "Action": [
       "nas:DescribeMountTargets",
       "nas:CreateMountTarget",
       "nas:DeleteFileSystem",
       "nas:DeleteMountTarget",
       "nas:CreateFileSystem"
     ],
     "Resource": [
       "*"
     ],
       "Effect": "Allow"
   }
   ```

   You can grant the permissions by using the following methods:

   - Attach the preceding permission policy to the master RAM role of your ACK cluster. For more information, see ACK default roles.



> **Note** The master RAM role is automatically assigned to a managed Kubernetes cluster. However, you must manually assign the master RAM role to a dedicated Kubernetes cluster.

- Create a RAM user and attach the preceding permission policy to the RAM user. Then, generate an AccessKey pair for the RAM user and specify the AccessKey pair in the `env` variable of csi-nasprovisioner in the configurations of the csi-provisioner StatefulSet. For more information, see ACK default roles.

```
env:
  - name: CSI_ENDPOINT
    value: unix://socketDir/csi.sock
  - name: ACCESS_KEY_ID
    value: ""
  - name: ACCESS_KEY_SECRET
    value: ""
```

2. Create a StorageClass.

i. Create an *alicloud-nas-fs.yaml* file and copy the following content into the file:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: alicloud-nas-fs
mountOptions:
- nolock,tcp,noresvport
- vers=3
parameters:
 volumeAs: filesystem
 vpcId: "vpc-xxxxxxxxxxx"
 vSwitchId: "vsw-xxxxxxxxx"
 deleteVolume: "false"
provisioner: nasplugin.csi.alibabacloud.com
reclaimPolicy: Retain
```

Parameter	Description
volumeAs	The mode in which the NAS file system is mounted. Supported modes are: ■ filesystem: csi-nasprovisioner automatically creates a NAS file system. Each PV corresponds to a separate NAS file system. ■ subpath: csi-nasprovisioner automatically creates a subdirectory in a NAS file system. Each PV corresponds to a separate subdirectory of a NAS file system.
storageType	The type of NAS file system. You can select **Performance** or **Capacity**. Default value: Performance.
zoneId	The ID of the zone to which the NAS file system belongs.
vpcId	The ID of the virtual private cloud (VPC) to which the mount target of the NAS file system belongs.
vSwitchId	The ID of the vSwitch to which the mount target of the NAS file system belongs.
accessGroupName	The permission group to which the mount target of the NAS file system belongs. Default value: DEFAULT_VPC_GROUP_NAME.
deleteVolume	The reclaim policy of a NAS file system when the related PV is deleted. NAS is a shared storage service. Therefore, you must specify both deleteVolume and reclaimPolicy to ensure data security.
reclaimPolicy	The policy for reclaiming the PV. The reclaim policy of a NAS file system. When you delete a PVC, the related NAS file system is automatically deleted only if you set deleteVolume to true and reclaimPolicy to Delete.

ii. Run the following command to create a StorageClass:

```
kubectl create -f alicloud-nas-fs.yaml
```

3. Create a PVC and pods to mount a NAS volume.

i. Create a *nas.yaml* file and copy the following content into the file:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: nas-csi-pvc-fs
spec:
 accessModes:
  - ReadWriteMany
 storageClassName: alicloud-nas-fs
 resources:
  requests:
   storage: 20Gi
```

ii. Create an *nginx.yaml* file and copy the following content into the file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: deployment-nas-fs
 labels:
  app: nginx
spec:
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    volumeMounts:
     - name: nas-pvc
      mountPath: "/data"
   volumes:
    - name: nas-pvc
     persistentVolumeClaim:
      claimName: nas-csi-pvc-fs
```

iii. Run the following command to create the PV and PVC:

```
kubectl create -f pvc.yaml -f nginx.yaml
```

In filesystem mode, the CSI driver automatically creates a NAS file system and a mount target when you create the PVC. When the PVC is deleted, the file system and the mount target are retained or deleted based on the settings of the deleteVolume and reclaimPolicy parameters.

Verify that the NAS file system can be used to persist data

1. Query the pods that run the applications and the mounted NAS file system.

i. Run the following command to query the pods that run the applications:

```
kubectl get pod
```

Expected output:

```
NAME                READY  STATUS   RESTARTS  AGE
deployment-nas-1-5b5cdb85f6-n****  1/1   Running  0     32s
deployment-nas-2-c5bb4746c-4****   1/1   Running  0     32s
```

ii. Run the following command to query files in the */data* path of a pod. The pod `deployment-nas-1-5b5cdb85f6-n****` is used as an example:

```
kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
```

No output is returned. This indicates that no file exists in the */data* path.

2. Run the following command to create a file named *nas* in the */data* path of the pod `deployment-nas-1-5b5cdb85f6-n****` :

```
kubectl exec deployment-nas-1-5b5cdb85f6-n**** touch /data/nas
```

3. Run the following command to query files in the */data* path of the pod `deployment-nas-1-5b5cdb85f6-n****` :

```
kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
```

Expected output:

```
nas
```

4. Run the following command to delete the pod:

```
kubectl delete pod deployment-nas-1-5b5cdb85f6-n****
```

5. Open another kubectl CLI and run the following command to query how the pod is deleted and recreated:

```
kubectl get pod -w -l app=nginx
```

6. Verify that the file still exists after the pod is deleted.

i. Run the following command to query the name of the recreated pod:

```
kubectl get pod
```

Expected output:

```
NAME                READY  STATUS   RESTARTS  AGE
deployment-nas-1-5b5cdb85f6-n****  1/1   Running  0     32s
deployment-nas-2-c5bb4746c-4****   1/1   Running  0     32s
```

ii. Run the following command to query files in the */data* path of the pod `deployment-nas-1-5b5cdb85f6-n****` :

```
kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
```

Expected output:

```
nas
```

The *nas* file still exists in the /data path. This indicates that data is persisted to the NAS file system.

Verify that the data in the NAS file system can be shared across pods

1. Query the pods that run the applications and the files in the mounted NAS file system.
   i. Run the following command to query the pods that run the applications:

   ```
   kubectl get pod
   ```

   Expected output:

   ```
   NAME                READY  STATUS   RESTARTS  AGE
   deployment-nas-1-5b5cdb85f6-n****  1/1   Running  0     32s
   deployment-nas-2-c5bb4746c-4****   1/1   Running  0     32s
   ```

   ii. Run the following command to query files in the */data* path of each pod:

   ```
   kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
   kubectl exec deployment-nas-2-c5bb4746c-4**** ls /data
   ```

2. Run the following command to create a file named *nas* in the */data* path of a pod:

   ```
   kubectl exec deployment-nas-1-5b5cdb85f6-n**** touch /data/nas
   ```

3. Run the following command to query files in the */data* path of each pod:
   i. Run the following command to query files in the */data* path of the pod `deployment-nas-1-5b5cdb85f6-n****`:

   ```
   kubectl exec deployment-nas-1-5b5cdb85f6-n**** ls /data
   ```

   Expected output:

   ```
   nas
   ```

   ii. Run the following command to query files in the */data* path of the pod `deployment-nas-2-c5bb4746c-4****`:

   ```
   kubectl exec deployment-nas-2-c5bb4746c-4**** ls /data
   ```

   Expected output:

   ```
   nas
   ```

   When you create a file in the */data* path of one pod, you can also find the file in the */data* path of the other pod. This indicates that data is the NAS file system is shared by the two pods.

8.5.4. Set quotas on the subdirectories of NAS volumes

You can set quotas to manage resource allocation and improve the overall resource utilization. Container Service for Kubernetes (ACK) allows you to use the CSI plug-in to set quotas on the subdirectories of Apsara File Storage NAS volumes. This topic describes how to set quotas on the subdirectories of NAS volumes.

Prerequisites

- The image version of csi-plugin is V1.18.8.45 or later. For more information about csi-plugin versions, see csi-plugin.
- The NAS volume is mounted by using a subdirectory.

Limits

- Only NAS Capacity file systems support quota limits. For more information about the types of NAS file systems, see General-purpose NAS file systems.
- Quotas can be set only for volumes that are mounted by using subdirectories.
- Quota limits are supported in all regions except the China (Hohhot) and China (Ulanqab) regions.
- For each file system, you can configure quotas only on a maximum of 10 directories.
  - You can set an enforcement quota on a directory. If the quota is exceeded, you cannot write data to the directory. The write operations include the operations that are used to increase the length of files, create files, subdirectories, and special files, and move files to another directory. An IOError error occurs at the frontend.
  - To avoid unexpected errors, use caution when you set enforcement quotas on critical directories.
  - A specific period of time is required before an enforcement quota is enabled or disabled due to asynchronous execution at the backend. In most cases, the time period ranges from 5 to 15 minutes.

Examples

1. Create a StorageClass that uses a subdirectory of a NAS file system to provision volumes.

   In this example, the following template is used:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: alicloud-nas-sp8
mountOptions:
  - nolock,tcp,noresvport
  - vers=3
parameters:
  volumeAs: subpath
  server: "xxx.cn-hangzhou.nas.aliyuncs.com:/"
  archiveOnDelete: "false"
  path: "/abc"
  volumeCapacity: "true"
provisioner: nasplugin.csi.alibabacloud.com
reclaimPolicy: Delete
allowVolumeExpansion: "true"
```

Parameter	Description
mountOptions	Set the options parameter and Network File System (NFS) version in the mountOptions field.
volumeAs	You can select subpath or filesystem. subpath specifies that a subdirectory is mounted on the cluster while filesystem specifies that a file system is mounted on the cluster.
server	When you mount a subdirectory of the NAS file system as a persistent volume (PV), this parameter specifies the mount target of the NAS file system.
archiveOnDelete	This parameter specifies whether to delete the backend storage when reclaimPolicy is set to Delete. NAS is a shared storage service. You must set both reclaimPolicy and deleteVolume to ensure data security. Default value: true.
path	The subdirectory of the NAS file system that is mounted. If you mount an Extreme NAS file system, the path must start with */share*.

Parameter	Description
volumeCapacity	This parameter specifies whether to set a quota. Valid values: true and false.
provisioner	The provisioner of the PV that is provided by ACK.
reclaimPolicy	The policy that is used to reclaim the PV. Valid values: ○ Retain: retains the backend storage when the PV and PVC are deleted. The backend storage may be cloud disks. ○ Delete: automatically deletes the backend storage and PV when the PVC is deleted.
allowVolumeExpansion	This parameter specifies whether NAS storage volume expansion is supported.

Note To create a StorageClass that sets quotas on the subdirectory of a NAS file system, the volumeCapacity parameter must be set to true.

2. Create a PVC that claims a storage capacity of 20 GiB.

   In this example, the following template is used:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: pvc-nas-dynamic-create-subpath8
spec:
 accessModes:
  - ReadWriteMany
 storageClassName: alicloud-nas-sp8
 resources:
  requests:
   storage: 20Gi
```

3. Create a Deployment that uses the PVC that is created in Step 2.

   In this example, the following template is used:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: deployment-nas-dynamic-create8
 labels:
  app: nginx
spec:
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
    - name: nginx
      image: nginx:1.14.2
      ports:
       - containerPort: 80
      volumeMounts:
       - name: pvc-nas-dynamic-create-subpath8
         mountPath: "/data"
   volumes:
    - name: pvc-nas-dynamic-create-subpath8
      persistentVolumeClaim:
       claimName: pvc-nas-dynamic-create-subpath8
```

Verification

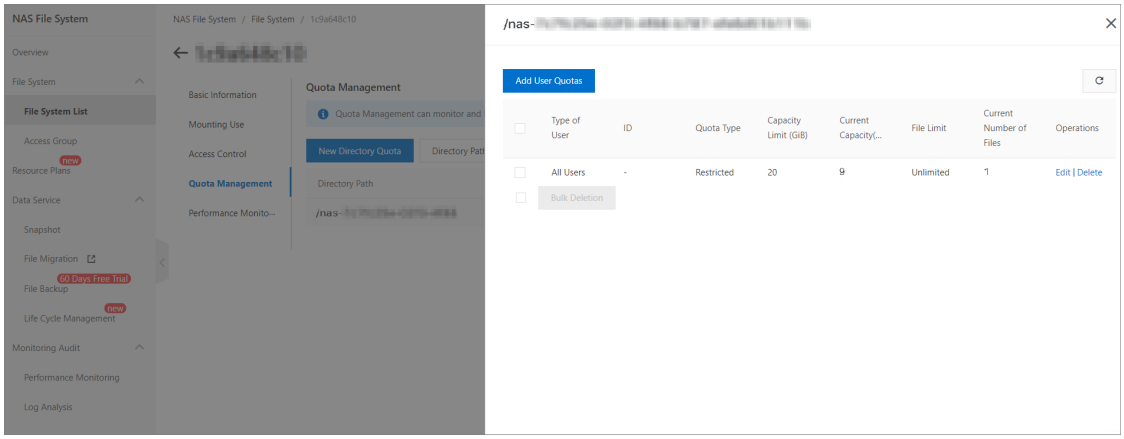
1. Run the following command to write 10 GiB of data to the */data* directory that is mounted on the Deployment that is created in Step 3:

   ```
   dd if=/dev/zero of=10G.txt bs=1M count=10000
   ```

2. Wait 5 to 15 minutes and then check the quota details of the subdirectory.
   i. Log on to the NAS console.
   ii. In the left-side navigation pane, choose **File System > File System List**.
   iii. Select the NAS file system that you have used and choose **More > Quota Management** in the **Operations** column.

iv. On the **Quota Management** page, click **Manage Quotas** in the **Operations** column.

The following figure shows that the subdirectory has a quota limit of 20 GiB. The used storage is 9 GiB.



When the 20 GiB of storage is used up, the `Disk quota exceeded` error appears if you try to write more data to the subdirectory.

```
sh-4.2# dd if=/dev/zero of=1G.txt bs=1M count=1000
dd: error writing '1G.txt': Disk quota exceeded
311+0 records in
310+0 records out
325058560 bytes (325 MB) copied, 1.01805 s, 319 MB/s
```

8.6. OSS volumes

8.6.1. Overview

You can mount Object Storage Service (OSS) buckets to Container Service for Kubernetes (ACK) clusters. Only statically provisioned OSS buckets can be mounted to ACK clusters.

Prerequisites

An OSS bucket is created in the OSS console. For more information, see Create buckets.

Instruction

The following section describes how to configure a statically provisioned OSS volume:

- An OSS bucket can be shared by multiple pods.
- bucket: You can mount only buckets to clusters. The subdirectories or files in a bucket cannot be mounted to an ACK cluster.
- url: the endpoint of an OSS bucket. If the bucket and the node to which the bucket is mounted are deployed in the same region, you can specify the internal endpoint of an OSS bucket.
- akId: your AccessKey ID.
- akSecret: your AccessKey secret.
- otherOpts: the custom parameters that are used to mount the OSS bucket. The parameters must be in the following format: `-o *** -o ***`.
- To mount an OSS volume, do not specify subpath.
- We recommend that you create a persistent volume (PV) for each application.

- Only the CentOS and Alibaba Cloud Linux 2 operating systems are supported.

Considerations

- OSS is a Filesystem in Userspace (FUSE) file system that can be mounted by using OSSFS. This method is suitable for read operations. For example, you can use this method to read configuration files, video files, and images.
- OSSFS is not suitable for write operations. If you require write operations, we recommend that you use Apsara File Storage NAS (NAS) file systems.
- Compared with FUSE, the file system in a kernel state offers higher stability and performance. We recommend that you use NAS file systems instead of OSS buckets in production environments.
- You can modify parameter configurations to optimize OSSFS performance in caching and permission management. For more information, see FAQ about OSSFS, ossfs/README-CN.md, and FAQ.

8.7. CPFS volumes

8.7.1. Static volumes

Cloud Paralleled File System (CPFS) is a parallel file system. CPFS stores data across multiple data nodes in a cluster and allows data to be simultaneously accessed by multiple clients. Therefore, CPFS can provide data storage services with high input/output operations per second (IOPS), high throughput, and low latency for large and high-performance computing clusters. This topic describes how to mount and use a CPFS volume in a Container Service for Kubernetes (ACK) cluster.

Context

ACK allows you to mount CPFS file systems as static or dynamic volumes. To mount CPFS file systems as static volumes, you must manually create persistent volumes (PVs) and persistent volume claims (PVCs). This procedure can be time-consuming when large numbers of PVs and PVCs are required. In this case, you can mount CPFS file systems as dynamic volumes. In this topic, alibaba-cloud-csi-driver is used to mount a CPFS file system. For more information, see alibaba-cloud-csi-driver.

> **Note** ACK supports only CPFS 1.0. CPFS 2.0 is not supported.

Procedure

1. Deploy the cpfs-plugin component.

   To mount CPFS volumes, you must first perform the following steps to deploy cpfs-plugin in your ACK cluster:

```
apiVersion: storage.k8s.io/v1
kind: CSIDriver
metadata:
 name: cpfsplugin.csi.alibabacloud.com
spec:
 attachRequired: false
 podInfoOnMount: true
---
# This YAML defines all API objects to create RBAC roles for csi node plugin.
kind: DaemonSet
apiVersion: apps/v1
metadata:
 name: csi-cpfsplugin
 namespace: kube-system
spec:
```

```
selector:
 matchLabels:
  app: csi-cpfsplugin
template:
 metadata:
  labels:
   app: csi-cpfsplugin
 spec:
  tolerations:
  - operator: Exists
  priorityClassName: system-node-critical
  serviceAccount: admin
  hostNetwork: true
  hostPID: true
  containers:
  - name: driver-registrar
   image: registry.cn-hangzhou.aliyuncs.com/acs/csi-node-driver-registrar:v1.2.0
   imagePullPolicy: Always
   args:
   - "--v=5"
   - "--csi-address=/var/lib/kubelet/csi-plugins/cpfsplugin.csi.alibabacloud.com/csi.sock"
   - "--kubelet-registration-path=/var/lib/kubelet/csi-plugins/cpfsplugin.csi.alibabacloud.com/csi.sock"
   volumeMounts:
   - name: kubelet-dir
    mountPath: /var/lib/kubelet/
   - name: registration-dir
    mountPath: /registration
  - name: csi-cpfsplugin
   securityContext:
    privileged: true
    capabilities:
     add: ["SYS_ADMIN"]
    allowPrivilegeEscalation: true
   image: registry.cn-hangzhou.aliyuncs.com/acs/csi-cpfsplugin:v1.16.6-d539237c
   imagePullPolicy: "Always"
   args:
   - "--endpoint=$(CSI_ENDPOINT)"
   - "--v=5"
   - "--driver=cpfsplugin.csi.alibabacloud.com"
   env:
   - name: CSI_ENDPOINT
    value: unix://var/lib/kubelet/csi-plugins/cpfsplugin.csi.alibabacloud.com/csi.sock
   volumeMounts:
   - name: kubelet-dir
    mountPath: /var/lib/kubelet/
    mountPropagation: "Bidirectional"
   - mountPath: /var/log/
    name: host-log
   - name: etc
    mountPath: /host/etc
  volumes:
  - name: kubelet-dir
   hostPath:
    path: /var/lib/kubelet/
    type: Directory
  - name: registration-dir
   hostPath:
```

```
      path: /var/lib/kubelet/plugins_registry
      type: DirectoryOrCreate
    - name: host-log
      hostPath:
        path: /var/log/
    - name: etc
      hostPath:
        path: /etc
  updateStrategy:
    type: RollingUpdate
```

Run the following command to check the state of cpfs-plugin:

```
# kubectl get pod -nkube-system |grep cpfs
csi-cpfsplugin-8t585                    2/2    Running  0      4h43m
csi-cpfsplugin-9z5xj                    2/2    Running  0      4h43m
csi-cpfsplugin-bdm22                    2/2    Running  0      4h43m
csi-cpfsplugin-bjnlx                    2/2    Running  0      4h44m
csi-cpfsplugin-nv7vg                    2/2    Running  0      4h43m
csi-cpfsplugin-zc7z5                    2/2    Running  0      4h43m
```

2. Use a CPFS volume.
   i. Create a CPFS file system. For more information, see Create a file system.
   ii. Create a PV and a PVC.

   Use the following template to create a PV and a PVC for the CPFS file system:

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: cpfs-csi-pv
 labels:
  alicloud-pvname: cpfs-pv
spec:
 capacity:
  storage: 5Gi
 accessModes:
  - ReadWriteOnce
 persistentVolumeReclaimPolicy: Retain
 csi:
  driver: cpfsplugin.csi.alibabacloud.com
  volumeHandle: cpfs-csi-pv
  volumeAttributes:
   server: "xxxxxx@tcp:xxxxxx@tcp"
   fileSystem: "xxxxxx"
   subPath: "/k8s"
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: cpfs-pvc
spec:
 accessModes:
  - ReadWriteOnce
 resources:
  requests:
   storage: 5Gi
 selector:
  matchLabels:
   alicloud-pvname: cpfs-pv
```

iii. Create a Deployment.

Use the following template to create a Deployment that uses the preceding PVC:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: deployment-cpfs
 labels:
  app: nginx
spec:
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    volumeMounts:
     - name: cpfs-pvc
       mountPath: "/data"
   volumes:
    - name: cpfs-pvc
      persistentVolumeClaim:
       claimName: cpfs-pvc
```

8.7.2. Dynamic volumes

Cloud Paralleled File System (CPFS) is a parallel file system. CPFS stores data across multiple data nodes in a cluster and allows data to be simultaneously accessed by multiple clients. Therefore, CPFS can provide data storage services with high input/output operations per second (IOPS), high throughput, and low latency for large and high-performance computing clusters. This topic describes how to mount and use a CPFS volume in a Container Service for Kubernetes (ACK) cluster.

Context

ACK allows you to mount CPFS file systems as static or dynamic volumes. To mount CPFS file systems as static volumes, you must manually create persistent volumes (PVs) and persistent volume claims (PVCs). This procedure can be time-consuming when large numbers of PVs and PVCs are required. In this case, you can mount CPFS file systems as dynamic volumes. In this topic, alibaba-cloud-csi-driver is used to mount a CPFS volume. For more information, see alibaba-cloud-csi-driver.

> **Note** ACK supports only CPFS 1.0. CPFS 2.0 is not supported.

Procedure

1. Deploy the cpfs-plugin component.

   To mount CPFS file systems, you must first take the following steps to deploy components cpfs-provisioner and cpfs-plugin in your ACK cluster:

   - Use the following template to deploy cpfs-provisioner:

```
kind: Deployment
apiVersion: apps/v1
metadata:
 name: csi-cpfs-provisioner
 namespace: kube-system
spec:
 selector:
  matchLabels:
   app: csi-cpfs-provisioner
 replicas: 2
 template:
  metadata:
   labels:
    app: csi-cpfs-provisioner
  spec:
   tolerations:
   - operator: "Exists"
   affinity:
    nodeAffinity:
     preferredDuringSchedulingIgnoredDuringExecution:
     - weight: 1
       preference:
         matchExpressions:
         - key: node-role.kubernetes.io/master
           operator: Exists
   priorityClassName: system-node-critical
   serviceAccount: admin
   hostNetwork: true
   containers:
    - name: external-cpfs-provisioner
      image: registry.cn-hangzhou.aliyuncs.com/acs/csi-provisioner:v1.4.0-aliyun
      args:
       - "--provisioner=cpfsplugin.csi.alibabacloud.com"
       - "--csi-address=$(ADDRESS)"
       - "--volume-name-prefix=cpfs"
       - "--timeout=150s"
       - "--enable-leader-election=true"
       - "--leader-election-type=leases"
       - "--retry-interval-start=500ms"
       - "--v=5"
      env:
       - name: ADDRESS
         value: /socketDir/csi.sock
      imagePullPolicy: "Always"
      volumeMounts:
       - name: socket-dir
         mountPath: /socketDir
   volumes:
    - name: socket-dir
      hostPath:
       path: /var/lib/kubelet/csi-plugins/cpfsplugin.csi.alibabacloud.com
       type: DirectoryOrCreate
```

- Deploy csi-plugin.

```
apiVersion: storage.k8s.io/v1
kind: CSIDriver
```

```
metadata:
 name: cpfsplugin.csi.alibabacloud.com
spec:
 attachRequired: false
 podInfoOnMount: true
---
# This YAML defines all API objects to create RBAC roles for csi node plugin.
kind: DaemonSet
apiVersion: apps/v1
metadata:
 name: csi-cpfsplugin
 namespace: kube-system
spec:
 selector:
  matchLabels:
   app: csi-cpfsplugin
 template:
  metadata:
   labels:
    app: csi-cpfsplugin
  spec:
   tolerations:
   - operator: Exists
   priorityClassName: system-node-critical
   serviceAccount: admin
   hostNetwork: true
   hostPID: true
   containers:
   - name: driver-registrar
    image: registry.cn-hangzhou.aliyuncs.com/acs/csi-node-driver-registrar:v1.2.0
    imagePullPolicy: Always
    args:
    - "--v=5"
    - "--csi-address=/var/lib/kubelet/csi-plugins/cpfsplugin.csi.alibabacloud.com/csi.sock"
    - "--kubelet-registration-path=/var/lib/kubelet/csi-plugins/cpfsplugin.csi.alibabacloud.com/csi.sock"
    volumeMounts:
    - name: kubelet-dir
     mountPath: /var/lib/kubelet/
    - name: registration-dir
     mountPath: /registration
   - name: csi-cpfsplugin
    securityContext:
     privileged: true
     capabilities:
      add: ["SYS_ADMIN"]
     allowPrivilegeEscalation: true
    image: registry.cn-hangzhou.aliyuncs.com/acs/csi-cpfsplugin:v1.16.6-d539237c
    imagePullPolicy: "Always"
    args:
    - "--endpoint=$(CSI_ENDPOINT)"
    - "--v=5"
    - "--driver=cpfsplugin.csi.alibabacloud.com"
    env:
    - name: CSI_ENDPOINT
     value: unix://var/lib/kubelet/csi-plugins/cpfsplugin.csi.alibabacloud.com/csi.sock
    volumeMounts:
     name: kubelet-dir
```

```
    - name: kubelet-dir
      mountPath: /var/lib/kubelet/
      mountPropagation: "Bidirectional"
    - mountPath: /var/log/
      name: host-log
    - name: etc
      mountPath: /host/etc
  volumes:
  - name: kubelet-dir
    hostPath:
      path: /var/lib/kubelet/
      type: Directory
  - name: registration-dir
    hostPath:
      path: /var/lib/kubelet/plugins_registry
      type: DirectoryOrCreate
  - name: host-log
    hostPath:
      path: /var/log/
  - name: etc
    hostPath:
      path: /etc
updateStrategy:
  type: RollingUpdate
```

2. Use a dynamic CPFS volume.
   i. Use the following template to create a StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: alicloud-cpfs
parameters:
  volumeAs: subpath
  server: "cpfs-0237cc41-**.cn-shenzhen.cpfs.nas.aliyuncs.com@tcp:cpfs-0237cc41-**.cn-shenzhen.cpfs.
nas.aliyuncs.com@tcp:/0237cc41"
  archiveOnDelete: "false"
provisioner: cpfsplugin.csi.alibabacloud.com
reclaimPolicy: Delete
```

Parameter	Description
volumeAs	The type of dynamic volume. Only the subpath type is supported. This type indicates that the volume is created from a subdirectory on the CPFS server.
server	The address of the CPFS server.
archiveOnDelete	Specifies how the CPFS subdirectory is handled when the corresponding PVC and PV are deleted. ■ If reclaimPolicy is set to Delete and archiveOnDelete is set to false, the subdirectory and its data are directly deleted. Exercise caution. ■ If reclaimPolicy is set to Delete and archiveOnDelete is set to true, the subdirectory is renamed and archived. ■ If reclaimPolicy is set to Retain, the subdirectory is retained.

ii. Use the following template to create a PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: cpfs-pvc
spec:
 accessModes:
  - ReadWriteMany
 storageClassName: alicloud-cpfs
 resources:
  requests:
   storage: 20Gi
```

> **Notice** Specify storageClassName in the PVC definition.

iii. Use the following template to create a Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: deployment-cpfs
 labels:
  app: nginx
spec:
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    volumeMounts:
     - name: cpfs-pvc
       mountPath: "/data"
   volumes:
    - name: cpfs-pvc
      persistentVolumeClaim:
       claimName: cpfs-pvc
```

8.8. AEP non-volatile storage volumes

8.8.1. Use AEP in ACK clusters

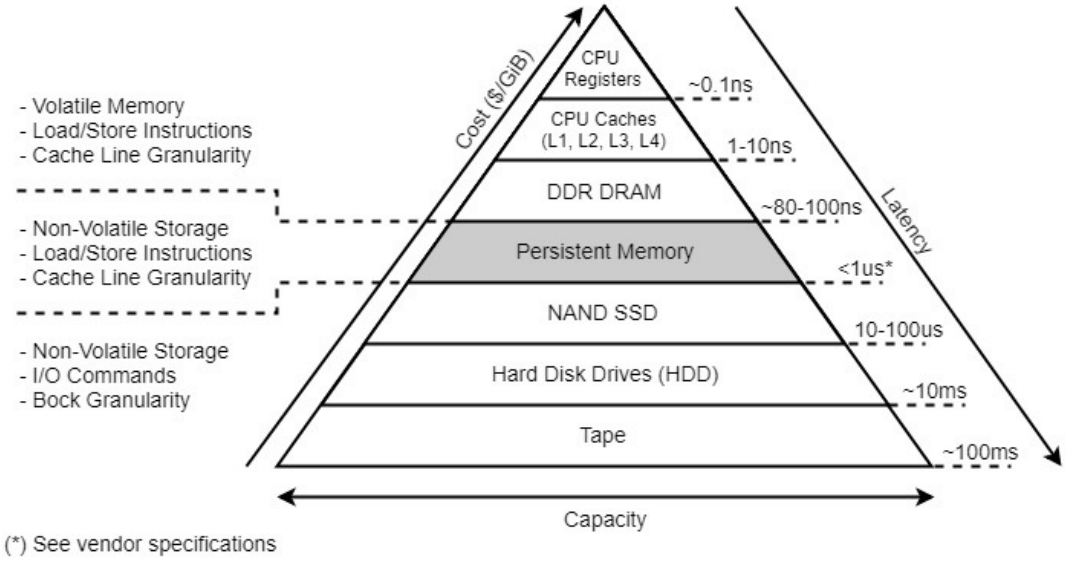
Traditional memory and storage products have limits on density, performance, and cost. Persistent memory (PMEM) is a storage product that provides high-performance, persistent, and in-memory storage services. Apache Pass (AEP) is a PMEM product developed by Intel. It integrates cutting edge memory and storage technologies. AEP expands memory capacity in a cost-effective manner and enables low-latency access to persisted data. You can use AEP in Container Service for Kubernetes (ACK) clusters. This topic describes how to use AEP in ACK clusters by using an example.

AEP overview

In recent years, CPU technology has rapidly developed. The increase in computing frequency of single-core CPUs and the emergence of multi-core CPUs have made a considerable contribution to the development of the computing capabilities of CPUs. However, storage media as data carriers have not kept up with the development of CPUs. Dynamic random-access memory (DRAM) provides high-speed reads and writes but loses data when power is off. In addition, DRAM does not apply to large-scale scenarios due to its high cost. NVM (such as SSDs and disks) does not lose data when power is off. However, they cannot provide high-speed reads and writes as required. Therefore, the computing industry has been in research for a memory product that can meet the computing requirements of CPUs and avoid data loss when power is off.

The release of AEP indicates that PMEM is becoming a mature technology. PMEM products provide high-speed reads and writes that are almost the same as those by using memory. PMEM products are being used in data acceleration industries.

The following figure shows the hierarchy of memory and storage media based on the cost, latency, and capacity. You can view the position of PMEM in the hierarchy.



How to use AEP in ACK

You can use the Container Storage Interface (CSI) driver provided by Alibaba Cloud to manage the lifecycle of AEP devices in ACK clusters. This allows you to allocate, mount, and use AEP resources through declarative claims.

You can use AEP in ACK clusters by using the following methods:

- **PMEP-LVM (use AEP as non-intrusive block storage)**
  You can directly claim the AEP resources without modifying your applications. You can use Logical Volume Manager (LVM) to virtualize PMEM resources on a node into volume groups (VGs). Then, you can create persistent volume claims (PVCs) of the required type and capacities. You can use AEP without modifying the following types of applications: serverless applications, low-latency and high-throughput data computing applications, and short-CI/CD period applications that require high-speed temporary storage. This allows you to improve the I/O throughput by 2 to 10 times.

- **PMEM-direct memory**
  You can use PMEM as direct memory by making a specific number of modifications to the memory allocation functions. This allows you to access data in almost the same way as using dynamic random access memory (DRAM). This way, you can provision AEP resources as direct memory at TB-level and reduce 30% to 50% of the cost. This meets the requirements of in-memory databases such as Redis and SAP HANA in terms of large memory and cost-effectiveness.

> **Note**
> PMEP-LVM: AEP resources can be used as block storage or file systems in ACK clusters without intrusion or modification to your applications. The I/O throughput is 2 to 10 times higher than SSDs.
> PMEM-direct memory: AEP resources can be used as direct memory in ACK clusters. You must modify the applications so that they are adaptive to the logic in PMEM SDK for memory allocation. This offers high throughput and low latency that are comparable to DRAM.

Comparison between PMEM and SSD

Method	Support for fragmented storage	Support for online expansion	Support for memory persistence	Application modifications	Latency (4K/RW)	Throughput (4K/RW)	Maximum capacity of a single ECS instance (ecs.ebmre6p.26xlarge)
PMEM-LVM	No	Yes	Yes	No	10 us	10W	1536GB
PMEM-Direct	Yes	No	No	Yes	1.2 us	56W	768GB
SSD	No	Yes	Yes	No	100 us	1W	32TB

Deploy the CSI plug-in for memory volumes

To use AEP in ACK clusters, you must deploy the following components:

- CSI-Plugin: initializes PMEM devices and creates, deletes, mounts, and unmounts volumes.
- CSI-Provisioner: detects and initiates volume creation and deletion requests.
- CSI-Scheduler: schedules storage (the ACK scheduler is preinstalled).

> **Note**
> When you deploy CSI-Plugin, take note of the following limits:
> - To enable automatic operations and maintenance (O&M) for AEP devices, you must add the `pmem.csi.alibabacloud.com` label to the node that uses AEP.
> - To use the PMEP-LVM method, you must add the `pmem.csi.alibabacloud.com: lvm` label to the node that uses AEP.
> - To use the PMEM-direct memory method, you must add the `pmem.csi.alibabacloud.com: direct` label to the node that uses AEP.

1. Create an ACK cluster.

   Create an ACK cluster that contains ECS instances with AEP resources, for example, ecs.ebmre6p.26xlarge. For more information about how to create an ACK cluster, see 创建Kubernetes托管版集群.

2. Configure the node with AEP resources.

   To enable the CSI plug-in to function as expected, you must add the required label to the node. Add the following label:

   ```
   pmem.csi.alibabacloud.com/type: direct
   ```

   or add the following label:

   ```
   pmem.csi.alibabacloud.com/type: lvm
   ```

3. Deploy the CSI plug-in for PMEM.

   a. Deploy CSI-Plugin. ❯
   b. Deploy CSI-Provisioner. ❯
   c. Create a StorageClass. ❯

Examples

1. **Use AEP as block storage volumes**

   i. Create a PVC.

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
    annotations:
     volume.kubernetes.io/selected-node: cn-zhangjiakou.192.168.1.12
    name: pmem-lvm
   spec:
    accessModes:
    - ReadWriteOnce
    resources:
     requests:
      storage: 10Gi
    storageClassName: pmem-lvm
   ```

   To schedule the PVC to a specific AEP node, add the following annotation to the PVC configurations: `annotations: volume.kubernetes.io/selected-node`.

ii. Deploy an application.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: deployment-lvm
 labels:
  app: nginx-lvm
spec:
 selector:
  matchLabels:
   app: nginx-lvm
 template:
  metadata:
   labels:
    app: nginx-lvm
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    command: ["sh", "-c"]
    args: ["sleep 10000"]
    volumeMounts:
     - name: pmem-pvc
       mountPath: "/data"
   volumes:
    - name: pmem-pvc
      persistentVolumeClaim:
       claimName: pmem-lvm
```

iii. Check the result.

```
kubectl get pvc
```

```
pmem-lvm      Bound   disk-334cb6fa-fabd-4687-96dc-0d2b15564822  10Gi    RWO       pmem-lvm
10m
```

```
kubectl get pod
```

```
NAME                  READY  STATUS   RESTARTS  AGE
deployment-lvm-5bf65c7f76-jb6nl   1/1   Running  0     10m
```

```
kubectl exec -ti deployment-lvm-5bf65c7f76-jb6nl sh
```

```
df /data
```

```
Filesystem                                    1K-blocks  Used Available Use% Mounted on
/dev/mapper/pmemvgregion0-disk--334cb6fa--fabd--4687--96dc--0d2b15564822  10255636 36888  10202
364  1% /data
```

The output shows that a block storage volume is created and mounted to the application pod.

1. **Use AEP as direct memory volumes**

i. Create a PVC.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 annotations:
  volume.kubernetes.io/selected-node: cn-zhangjiakou.192.168.1.14
 name: pmem-direct
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 9Gi
 storageClassName: pmem-direct
```

To schedule the PVC to a node with AEP resources, add the following annotation to the PVC configurations: `annotations: volume.kubernetes.io/selected-node`.

ii. Deploy an application.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: deployment-direct
 labels:
  app: nginx-direct
spec:
 selector:
  matchLabels:
   app: nginx-direct
 template:
  metadata:
   labels:
    app: nginx-direct
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    command: ["sh", "-c"]
    args: ["sleep 1000"]
    volumeMounts:
     - name: pmem-pvc
       mountPath: "/data"
   volumes:
    - name: pmem-pvc
      persistentVolumeClaim:
       claimName: pmem-direct
```

iii. Check the result.

```
kubectl get pvc pmem-direct
```

```
NAME      STATUS  VOLUME                  CAPACITY  ACCESS MODES  STORAGECLASS  AGE
pmem-direct  Bound  disk-15c75a5d-469e-4dd7-b67e-6bdc489288a2  9Gi    RWO      pmem-direct  1
7m
```

```
kubectl get pod
```

```
NAME                READY  STATUS   RESTARTS  AGE
deployment-direct-64d448d96-9znfk  1/1   Running  0     17m
```

```
kubectl exec -ti deployment-direct-64d448d96-9znfk sh
```

```
df /data
```

```
Filesystem    1K-blocks  Used Available Use% Mounted on
/dev/pmem0     9076344 36888  9023072  1% /data
```

The output shows that a PMEM volume is created and mounted to the application pod.

8.8.2. Use AEP non-volatile memory to improve read and write performance

Container Service for Kubernetes (ACK) allows you to use Apache Pass (AEP) non-volatile memory as block storage or file systems without modifications to your applications. This makes it easy to use AEP and improves read and write performance by 2 to 10 times compared with solid-state drives (SSDs). This topic describes how to use AEP non-volatile memory as volumes in ACK clusters to meet the requirements of data-intensive workloads and scenarios that require low-latency and high-speed temporary storage.

Scenario 1: Low-latency and high-speed temporary storage

The following scenarios require low-latency and high-speed temporary storage:

- Temporary storage in continuous integration and continuous delivery (CD/CD)
- Small file traversal
- High-throughput logging
- Temporary file reads and writes in serverless scenarios

Use FIO to test read and write performance

In this topic, two persistent volume claims (PVCs) are created based on AEP PMEM-LVM and SSD to test read and write performance.

1. Deploy the following YAML file to create two PVCs based on PMEM-LVM and SSD:

   The YAML definition of the PMEM-LVM PVC | The YAML definition of the SSD PVC

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: csi-pmem-lvm
mountOptions:
- dax
parameters:
 lvmType: striping
 nodeAffinity: "true"
 volumeType: PMEM
provisioner: localplugin.csi.alibabacloud.com
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: pmem-lvm-pvc
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 100Gi
 storageClassName: csi-pmem-lvm
```

2. Deploy the following YAML file to mount the volumes to pods and run FIO tests:

Mount the PMEM-LVM volume and run FIO tests | Mount the SSD volume and run FIO tests

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: deployment-pmem-lvm
 labels:
  app: pmem-lvm
spec:
 selector:
  matchLabels:
   app: pmem-lvm
 template:
  metadata:
   labels:
    app: pmem-lvm
  spec:
   containers:
   - name: fio-test
    image: registry.cn-hangzhou.aliyuncs.com/eric-dev/sysbench:fio
    command: ["sh", "-c"]
    args: ["sleep 10000"]
    volumeMounts:
     - name: pmem
      mountPath: "/data"
   volumes:
    - name: pmem
     persistentVolumeClaim:
      claimName: pmem-lvm-pvc
```

- Use the `kubectl exec` command to run FIO tests inside containers to test the write performance of the PMEM-LVM volume:

  ```
  mount | grep csi
  cd /data
  fio -filename=./testfile -direct=1 -iodepth 1 -thread -rw=randwrite -ioengine=psync -bs=4k -size=10G -numj
  obs=50 -runtime=180 -group_reporting -name=rand_100write_4k
  ```

  Expected output:

  ```
  write: IOPS=92.0k, BW=363MiB/s (381MB/s)(8812MiB/24262msec)
  lat (nsec): min=2054, max=95278, avg=10544.00, stdev=1697.17
  ```

- Use the `kubectl exec` command to run FIO tests inside containers to test the write performance of the SSD volume:

  ```
  cd /data
  fio -filename=./testfile -direct=1 -iodepth 1 -thread -rw=randwrite -ioengine=psync -bs=4k -size=10G -numj
  obs=50 -runtime=180 -group_reporting -name=rand_100write_4k
  ```

  Expected output:

  ```
  lat (usec): min=20, max=7168, avg=24.76, stdev=13.97
  write: IOPS=37.3k, BW=146MiB/s (153MB/s)(8744MiB/60001msec)
  ```

The output indicates that the read and write performance of the AEP device is two to three times higher than that of the SSD device.

Volume type	IOPS	Throughput
PMEM-LVM	92000	381 MB/s
SSD	37000	153 MB/s

Scenario 2: Data-intensive workloads

In this topic, an SSD and an AEP device are used to deploy MySQL databases. Then, the database write performance is tested.

Use an SSD to deploy a MySQL database

1. Create an SSD volume.
   i. Run the following command to create an SSD volume by using the Container Storage Interface (CSI) driver provided by Alibaba Cloud:

   ```
   kubectl apply -f disk-mysql.yaml
   ```

   In this example, the following *disk-mysql.yaml* template is used:

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
    name: disk-mysql
   spec:
    accessModes:
    - ReadWriteOnce
    resources:
     requests:
      storage: 100Gi
    storageClassName: alicloud-disk-topology
   ```

   ii. Run the following command to query the SSD volume:

   ```
   kubectl get pvc disk-mysql
   ```

   Expected output:

   ```
   NAME     STATUS  VOLUME    CAPACITY  ACCESS MODES  STORAGECLASS       AGE
   disk-mysql  Bound  d-***    100Gi    RWO       alicloud-disk-topology  10h
   ```

2. Create a MySQL instance based on the SSD volume.
   i. Run the following command to create a MySQL instance:

   ```
   kubectl apply -f mysql-normal-ssd.yaml
   ```

   In this example, the following *mysql-normal-ssd.yaml* template is used:

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
 name: mysql-normal
 labels:
  app: wordpress
spec:
 selector:
  matchLabels:
   app: wordpress
   tier: mysql
 strategy:
  type: Recreate
 template:
  metadata:
   labels:
    app: wordpress
    tier: mysql
  spec:
   hostNetwork: true
   containers:
   - image: mysql:5.6
    name: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
     valueFrom:
      secretKeyRef:
       name: mysql-pass
       key: password
    ports:
    - containerPort: 3306
     name: mysql
    volumeMounts:
    - name: mysql
     mountPath: /var/lib/mysql
   volumes:
    - name: mysql
     persistentVolumeClaim:
      claimName: disk-mysql
---
apiVersion: v1
kind: Secret
metadata:
 name: mysql-pass
type: Opaque
data:
 username: YWRtaW4=
 password: YWRtaW4=
```

ii. Run the following command to query the pod that runs the MySQL instance:

```
kubectl get pod | grep mysql-normal
```

Expected output:

```
mysql-normal-***    1/1   Running  0     10h
```

Use AEP to deploy a MySQL database

1. Create a PMEM-LVM volume.
   i. Run the following command to create a PMEM-LVM volume by using the CSI driver provided by Alibaba Cloud:

   ```
   kubectl apply -f csi-pmem-lvm.yaml
   ```

   In this example, the following *csi-pmem-lvm.yaml* template is used:

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
    name: pmem-mysql
   spec:
    accessModes:
    - ReadWriteOnce
    resources:
     requests:
      storage: 100Gi
    storageClassName: csi-pmem-lvm
   ```

   ii. Run the following command to query the PMEM-LVM volume:

   ```
   kubectl get pvc pmem-mysql
   ```

   Expected output:

   ```
   NAME      STATUS  VOLUME    CAPACITY  ACCESS MODES  STORAGECLASS      AGE
   pmem-mysql  Bound  d-***     100Gi    RWO       csi-pmem-lvm      10h
   ```

2. Create a MySQL instance based on the PMEM-LVM volume.
   i. Run the following command to create a MySQL instance based on the PMEM-LVM volume:

   ```
   kubectl apply -f mysql-normal-pmem.yaml
   ```

   In this example, the following *mysql-normal-pmem.yaml* template is used:

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
 name: mysql-normal
 labels:
  app: wordpress
spec:
 selector:
  matchLabels:
   app: wordpress
   tier: mysql
 strategy:
  type: Recreate
 template:
  metadata:
   labels:
    app: wordpress
    tier: mysql
  spec:
   hostNetwork: true
   nodeName: <NODE name of AEP worker node>
   containers:
   - image: mysql:5.6
    name: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
     valueFrom:
      secretKeyRef:
       name: mysql-pass
       key: password
    ports:
    - containerPort: 3306
     name: mysql
    volumeMounts:
    - name: mysql
     mountPath: /var/lib/mysql
   volumes:
    - name: mysql
     persistentVolumeClaim:
      claimName: pmem-mysql
---
apiVersion: v1
kind: Secret
metadata:
 name: mysql-pass
type: Opaque
data:
 username: YWRtaW4=
 password: YWRtaW4=
```

ii. Run the following command to query the pod that runs the MySQL instance:

```
kubectl get pod | grep mysql-normal
```

Expected output:

```
mysql-normal-***    1/1   Running  0     10h
```

Test database write performance

Run the following command on the pod to test database performance:

```
sysbench /root/sysbench/point_select.lua run --db-driver=mysql --report-interval=1
--mysql-table-engine=innodb --oltp-table-size=10000 --oltp-tables-count=32 --oltp-test-mode=complex
--time=100 --mysql-host=127.0.0.1 --mysql-port=3306 --mysql-user=root --mysql-password=admin
--mysql-db=sbtest --oltp-read-only=on --oltp-simple-ranges=0 --oltp-sum-ranges=0
--oltp-order-ranges=0 --oltp-distinct-ranges=0  --oltp-dist-type=uniform --warmup-time=300
--max-time=30 --max-requests=0 --percentile=99 --num-threads=150
```

Test result:

Volume type	Insert IOPS
SSD	49812
PMEM-LVM	122156

The result indicates that the database write performance is 2.5 times higher when the AEP device is used than when the SSD device is used.

Related information

- Use AEP in ACK clusters

8.8.3. Use AEP as direct memory to deploy a Redis database

Container Service for Kubernetes allows you to use persistent memory (PMEM) as direct memory by modifying the memory allocation functions. This offers high throughput and low latency that are comparable to dynamic random-access memory (DRAM). This topic describes how to use DRAM and PMEM to deploy Redis in-memory databases that require large memory capacity.

Context

When you use AEP as direct memory to deploy Redis in-memory databases, the following benefits are provided:

- Simplifies the use of AEP resources
- Saves the memory cost by 30% to 50%
- Minimizes application modifications

Redis uses AEP devices as direct memory and calls the MMAP function to map the consecutive address space of PMEM. This scenario requires specific Redis versions. You can find the Redis image address in the following examples.

Use DRAM to deploy a Redis in-memory database

DRAM provides high-speed reads and writes but loses data when power is off. DRAM does not apply to large-scale scenarios due to its high costs.

1. Run the following command to deploy a Redis instance:

```
kubectl apply -f redis-normal.yaml
```

In this example, the following *redis-normal.yaml* template is used:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: redis-normal
 labels:
  app: redis-normal
spec:
 selector:
  matchLabels:
   app: redis-normal
 template:
  metadata:
   labels:
    app: redis-normal
  spec:
   containers:
   - name: redis
    image: registry.cn-hangzhou.aliyuncs.com/plugins/redis:v1-normal
    imagePullPolicy: Always
   resources:
    requests:
     memory: 30Gi
    limits:
     memory: 30Gi
```

2. Run the following command to query the pod that runs the Redis instance:

```
kubectl get pod | grep redis-normal
```

Expected output:

```
redis-normal-***     1/1   Running  0      4d8h
```

3. Run the following command to test the write performance of Redis:

```
LD_PRELOAD=/usr/local/lib/libmemkind.so redis-benchmark -d 102400 -t set -n 1000000 -r 1000000
```

Expected output:

```
1000000 requests completed in 13.03 seconds
76751.86 requests per second
```

The preceding output shows that the Redis in-memory database that is deployed with DRAM can process approximately 76,751 requests per second.

Use PMEM to deploy a Redis in-memory database

PMEM provides non-volatile memory and has nearly the same access method and speed of DRAM.

1. Run the following command to deploy a Redis instance:

```
kubectl apply -f redis-normal.yaml
```

In this example, the following *redis-normal.yaml* template is used:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: pmem-pvc-direct
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 60Gi
 storageClassName: csi-pmem-direct
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: redis-pmem-direct
 labels:
  app: redis-pmem-direct
spec:
 selector:
  matchLabels:
   app: redis-pmem-direct
 template:
  metadata:
   labels:
    app: redis-pmem-direct
  spec:
   containers:
   - name: nginx
    image: registry.cn-hangzhou.aliyuncs.com/plugins/redis:v1-pmem
    imagePullPolicy: Always
    volumeMounts:
     - name: pmem-pvc
      mountPath: "/mnt/pmem"
   volumes:
    - name: pmem-pvc
     persistentVolumeClaim:
      claimName: pmem-pvc-direct
```

2. Run the following command to query the pod that runs the Redis instance:

```
kubectl get pod | grep redis-pmem-direct
```

Expected output:

```
redis-pmem-direct-5b469546db-5tk2j  1/1   Running  0      4d9h
```

3. Run the following command to test the write performance of Redis:

```
LD_PRELOAD=/usr/local/lib/libmemkind.so redis-benchmark -d 10240 -t set -n 1000000 -r 1000000
```

Expected output:

```
1000000 requests completed in 19.89 seconds
50266.41 requests per second
```

The preceding output shows that the Redis in-memory database that is deployed with PMEM can process about 50,266 requests per second.
Compared with the Redis in-memory database deployed with DRAM, the Redis in-memory database deployed with PMEM reduces the write performance by 34% but offers the same read performance and saves cost by 50%. ECS instances that use AEP devices, such as ecs.ebmre6p.26xlarge, have a maximum capacity of 1.5 TB PEM memory, which is five times the capacity of common in-memory databases.

8.9. Container Storage Monitoring

8.9.1. Use csi-plugin to monitor the storage resources of an ACK cluster

Container Service for Kubernetes (ACK) allows you to monitor disks and Apsara File Storage NAS (NAS) file systems that are used in ACK clusters by using Prometheus Service. This topic describes how to monitor disks and NAS file systems used in ACK clusters and provides an example on how to configure alert rules.

Prerequisites

Only csi-plugin of v1.18.8.46-afb19e46-aliyun can be used to monitor both disks and NAS file systems. Make sure that the YAML file and the image of csi-plugin are upgraded to the latest version. You can upgrade csi-plugin to the latest version by using the following methods:

- Download and run the upgrade script from the GitHub community to upgrade csi-plugin to the latest version. For more information, see alibaba-cloud-csi-driver.
- Upgrade csi-plugin by using the ACK console. For more information, see Upgrade csi-plugin by using the ACK console.

Context

In most cases, ACK clusters store data in disks or NAS file systems. You can install the latest version of csi-plugin in ACK clusters to map the drive letters of disks to the persistent volume claims (PVCs) that are used to mount the disks. You can use Prometheus Service to monitor the status of PVCs. You can also configure threshold-based alert rules to monitor the storage usage and I/O operations per second (IOPS). If the specified thresholds are reached, alerts are triggered.

Upgrade csi-plugin by using the ACK console

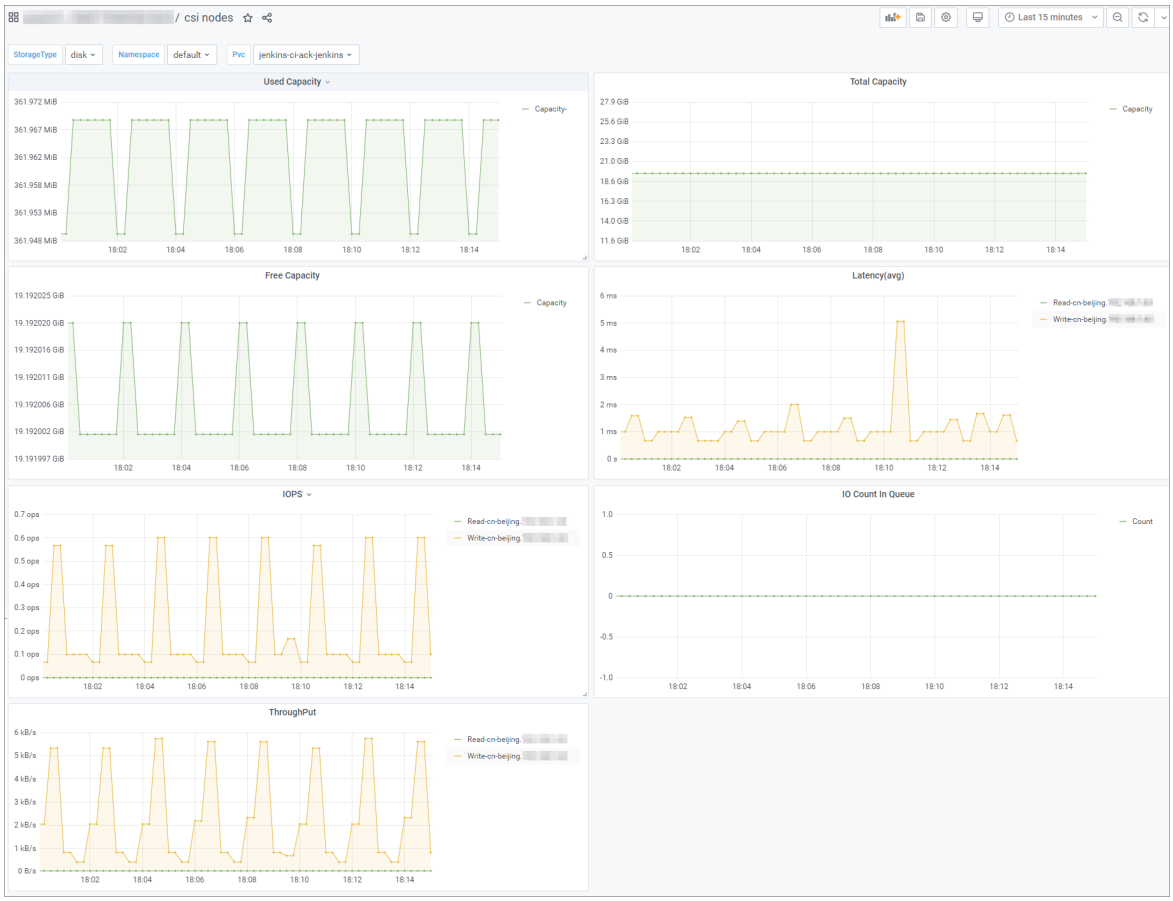
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. Click the **Storage** tab, find **csi-plugin** and click **Upgrade**.

How to monitor disks

Only disks and NAS file systems can be monitored at the node side. To view the monitoring data that is provided by Prometheus Service, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.

3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the cluster details page, choose **Operations > Prometheus Monitoring**.
5. Click the dashboard named **CSI Nodes**. In the dashboard that appears, you can view the monitoring data that is collected by csi-plugin from the nodes.
   The following figure shows an example of the dashboard.



The following tables describe the filters and metrics on the **CSI Nodes** dashboard.

- Filters

Filter	Description
StorageType	The volume type of the PVC.
Namespace	The namespace to which the PVC belongs.
Pvc	The name of the PVC.

- Metrics

Metric	Description	Unit
IOPS	The read/write IOPS of the PVC.	ops/s (number of reads/writes per second)

Metric	Description	Unit
Latency(avg)	The average latency at which the PVC processes read/write requests.	ms (milliseconds)
ThroughPut	The throughput of the PVC.	GB
IO Count In Queue	The number of I/O requests in the queue.	Count
Used Capacity	The used storage of the PVC.	GB
Total Capacity	The storage capacity of the PVC.	GB
Free Capacity	The available storage of the PVC.	GB

Examples on how to configure alert rules for disks

By default, alerting is disabled for disks that are mounted to ACK clusters. To enable alerting for disks, find the *csi-plugin.yaml* file by using the kubectl client and set the required parameters. For more information about how to use kubectl, see Connect to Kubernetes clusters by using kubectl.

In this example, the read/write latency of the disk is 10 milliseconds and the desired storage usage threshold is 85%. Set DISK_LATENCY_THRESHOLD to *10 ms* and DISK_CAPACITY_THRESHOLD_PERCENTAGE to *85%* in the env section of the *csi-plugin.yaml* file.

```
- name: DISK_LATENCY_THRESHOLD
  value: 10ms
- name: DISK_CAPACITY_THRESHOLD_PERCENTAGE
  value: 85%
```

The following code block shows some of the configurations in the *csi-plugin.yaml* file:

```
- name: csi-plugin
  securityContext:
    privileged: true
    capabilities:
      add: ["SYS_ADMIN"]
  allowPrivilegeEscalation: true
  image: registry.cn-hangzhou.aliyuncs.com/acs/csi-plugin:v1.18.8.45-1c5d2cd1-aliyun
  imagePullPolicy: "Always"
  args:
    - "--endpoint=$(CSI_ENDPOINT)"
    - "--v=2"
    - "--driver=oss,nas,disk"
  env:
    - name: KUBE_NODE_NAME
      valueFrom:
        fieldRef:
          apiVersion: v1
          fieldPath: spec.nodeName
    - name: CSI_ENDPOINT
      value: unix://var/lib/kubelet/csi-plugins/driverplugin.csi.alibabacloud.com-replace/csi.sock
    - name: MAX_VOLUMES_PERNODE
      value: "15"
    - name: SERVICE_TYPE
```

```
    value: "plugin"
  - name: DISK_LATENCY_THRESHOLD
    value: 10ms
  - name: DISK_CAPACITY_THRESHOLD_PERCENTAGE
    value: 85%
 livenessProbe:
  httpGet:
    path: /healthz
    port: healthz
    scheme: HTTP
  initialDelaySeconds: 10
  periodSeconds: 30
  timeoutSeconds: 5
  failureThreshold: 5
 ports:
  - name: healthz
    containerPort: 11260
    protocol: TCP
 volumeMounts:
  - name: kubelet-dir
    mountPath: /var/lib/kubelet/
    mountPropagation: "Bidirectional"
  - name: etc
    mountPath: /host/etc
  - name: host-log
    mountPath: /var/log/
  - name: ossconnectordir
    mountPath: /host/usr/
  - name: container-dir
    mountPath: /var/lib/container
    mountPropagation: "Bidirectional"
  - name: host-dev
    mountPath: /dev
    mountPropagation: "HostToContainer"
```

After you run the fio command to perform a stress test on the disk and the specified thresholds are reached, you can check the alerts by using the following methods:

- Run the `kubectl get events` command to check the triggered alerts.

```
14m         Warning   LatencyTooHigh       persistentvolumeclaim/pvc-disk-dynamic-create4   Pvc pvc-disk-dynamic-create4 latency load is too high, nodeName: cn-beijing.192.168.0.40, namespace: default, la
tency:22.38 ms, threshold:10.00 ms
13m         Warning   NotEnoughDiskSpace   persistentvolumeclaim/pvc-disk-dynamic-create4   Pvc pvc-disk-dynamic-create4 is not enough disk space, nodeName:cn-beijing.192.168.0.40, namespace: default, use
dPercentage:91.52%, threshold:85.00%
```

- Check the alerts in the ACK console.
  i. Log on to the ACK console.
  ii. In the left-side navigation pane, click **Clusters**.
  iii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
  iv. In the left-side navigation pane of the details page, choose **Operations > Events**. On the **Events** tab, you can view details about the alerts.

Warning	PersistentVolumeClaim pvc-disk-dynamic-create4	Pvc pvc-disk-dynamic-create4 is not enough disk space, nodeName:cn-beijing.[redacted], namespace: default, usedPercentage:91.52%, threshold:85.00%	NotEnoughDiskSpace	2021-02-24 16:31:10
Warning	PersistentVolumeClaim pvc-disk-dynamic-create4	Pvc pvc-disk-dynamic-create4 latency load is too high, nodeName: cn-beijing.[redacted], namespace: default, latency:22.38 ms, threshold:10.00 ms	LatencyTooHigh	2021-02-24 16:29:38

8.9.2. Use storage-operator to monitor the storage resources of an ACK cluster

You can deploy storage-operator in a Container Service for Kubernetes (ACK) cluster. After storage-operator is deployed, you can view the status of persistent volumes (PVs) and persistent volume claims (PVCs), the trend of PV and PVC status, and the PVs and PVCs that are in the abnormal state. This topic describes how to monitor the storage resources of an ACK cluster by using storage-operator.

Prerequisites

An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.

Step 1: Install storage-operator

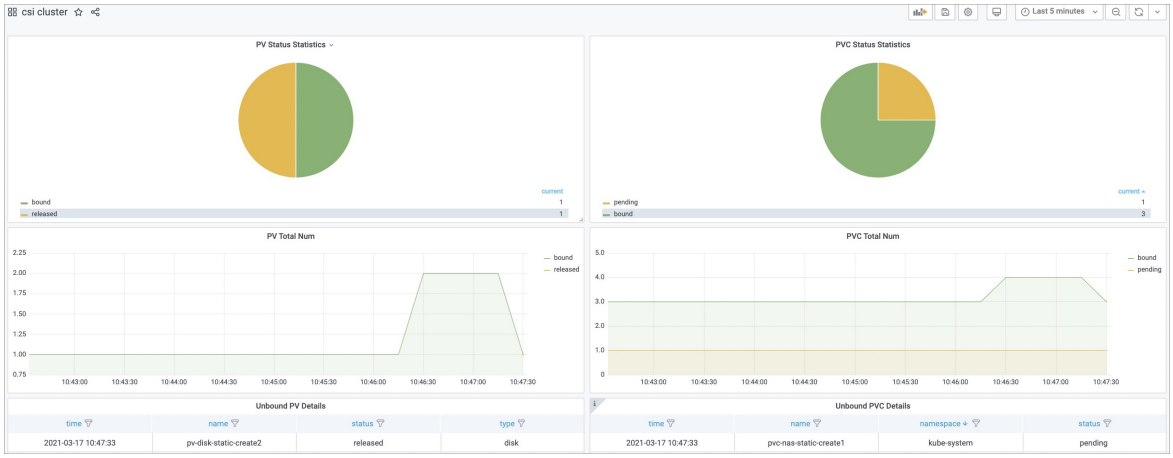
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. Click the **Storage** tab, find storage-operator, and then click **Install**.

Step 2: View monitoring data

Only PVs and PVCs can be monitored. You can view the monitoring data in Prometheus Service.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Prometheus Monitoring**.

   If storage-operator is not installed, click **Install** to install the component.
5. Click the **CSI Cluster** tab to go to the monitoring dashboards of the CSI plug-in. You can view the monitoring data in the dashboards.



The following table describes the metrics displayed in the dashboards on the CSI Cluster tab.

Metric	Description	Dimension
PV Status Statistics	Distribution of PV status.	Quantity and proportion.

Metric	Description	Dimension
PVC Status Statistics	Distribution of PVC status.	Quantity and proportion.
PV Total Num	Trend of PV status.	Quantity.
PVC Total Num	Trend of PVC status.	Quantity.
Unbound PV Details	Detailed information about PVs that are not in the Bound state. You can filter data by keyword.	Information is displayed in the following dimensions: time, name, status, and type.
Unbound PVC Details	Detailed information about PVCs that are not in the Bound state. You can filter data by keyword.	Information is displayed in the following dimensions: time, name, namespace, status.

8.10. Container Storage O&M

8.10.1. Use Storage Operator to deploy and upgrade storage components

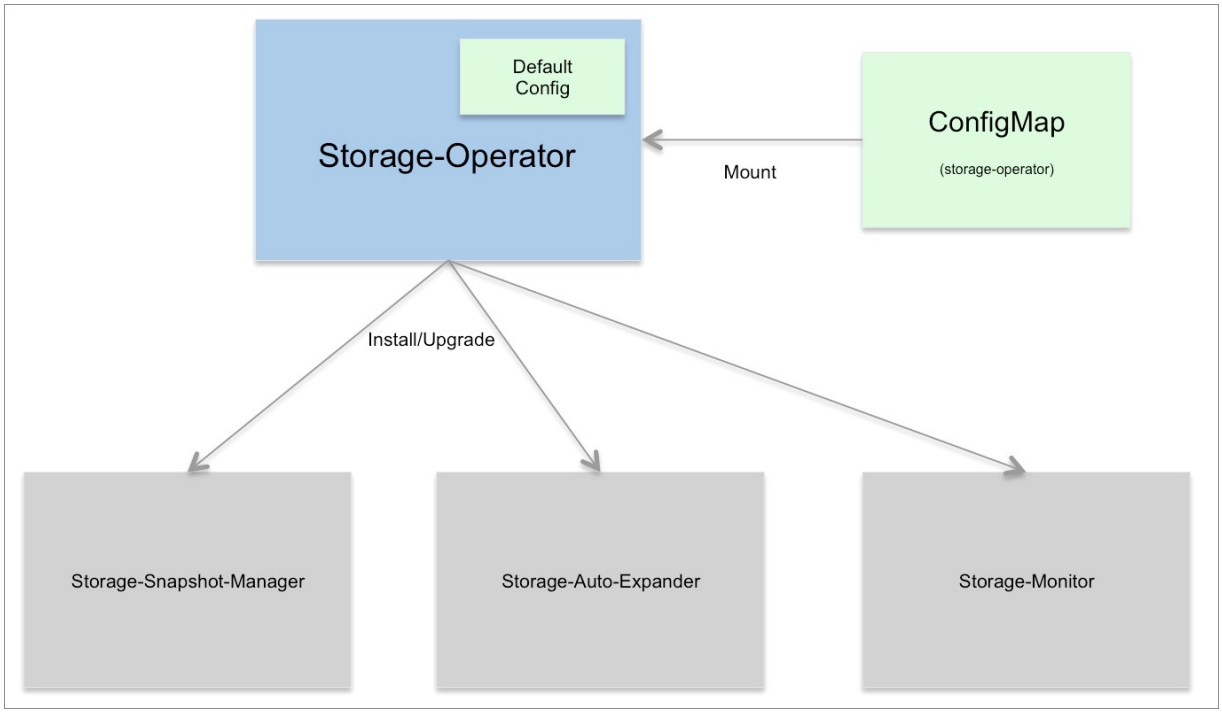
You can use Storage Operator to deploy and upgrade the storage components provided by Container Service for Kubernetes (ACK). You can use ConfigMaps to modify the configurations of storage components. This topic describes how to configure and deploy Storage Operator.

Introduction

Storage Operator is used to manage the lifecycle of storage components. Storage Operator runs as a Deployment, which deploys and upgrades storage components based on the default configurations inherited from the image and the custom configurations provided by ConfigMaps. This helps reduce the complexity of container development and maintenance.

- Default configurations: Storage Operator provides the default configurations of storage components. The default configurations vary based on the version of Storage Operator.
- Custom configurations: ConfigMaps can be used to define custom configurations of storage components, such as version information, and whether to install the component.

Storage Operator preferably uses custom configurations. The default configurations are used only when the custom configurations are not specified.



Note

- Each image of Storage Operator contains the default configuration file.
- When Storage Operator runs as a Deployment, the Deployment reads configurations from a ConfigMap file that is mounted on the Deployment and contains configurations of storage components.
- Storage Operator determines whether to deploy and upgrade a storage component by combining the default and custom configurations.

Step 1: Configure the components

Method 1: Use the default configurations provided by the image

ACK supports the following storage components: storage-snapshot-manager, storage-analyzer, storage-auto-expander, and storage-monitor.

Storage Operator provides the default configurations of storage components. The default configurations vary based on the version of Storage Operator. In this example, V1.18.8.0 is used.

```
{
 "storage-snapshot-manager":{
  "install": "true",
  "imageTag": "v1.18.8.0-81508da-aliyun",
  "imageRep": "acs/storage-snapshot-manager",
  "template": "/acs/templates/storage-snapshot-manager/install.yaml"
 },
 "storage-analzyer":{
  "install": "false",
  "imageTag": "v1.18.8.0-06c5560-aliyun",
  "imageRep": "acs/storage-analyzer",
  "template": "/acs/templates/storage-analyzer/install.yaml"
 },
 "storage-auto-expander":{
  "install": "false",
  "imageTag": "v1.18.8.0-4852fd4-aliyun",
  "imageRep": "acs/storage-auto-expander",
  "crdTmpl": "/acs/templates/storage-auto-expander/crd.yaml",
  "template": "/acs/templates/storage-auto-expander/install.yaml"
 },
 "storage-monitor":{
  "install": "true",
  "imageTag": "v1.18.8.0-c4744b6-aliyun",
  "imageRep": "acs/storage-monitor",
  "template": "/acs/templates/storage-monitor/install.yaml",
  "svcTmpl": "/acs/templates/storage-monitor/service.yaml"
 }
}
```

The following table describes the parameters in the default configurations.

Parameter	Description
install	Specifies whether to deploy the component. Default value: `false`.
imageTag	Specifies the default image version if the component is to be deployed.
imageRep	Specifies the default image name if the component is to be deployed.
crdTmpl	Specifies the template path of the Custom Resource Definition (CRD) that you want to deploy. If this parameter is empty, it indicates that no CRD is to be deployed.
svcTmpl	Specifies the template path of the Service that you want to deploy.
template	Specifies the template path of the component that you want to deploy.

Note You can use a ConfigMap to modify the parameters.

Method 2: Use a ConfigMap to customize the configurations

In this example, the following ConfigMap template is provided:

```
kind: ConfigMap
apiVersion: v1
metadata:
 name: storage-operator
 namespace: kube-system
data:
 storage-snapshot-manager: |
  # deploy config
  install:  false
  imageTag:  v1.16.aaaa
  imageRep:  acs/storage-snapshot-manager
  template:  /acs/templates/storage-snapshot-manager/install.yaml
  # env config
  SNAPSHOT_INTERVAL: 30
 storage-analyzer: |
  # deploy config
  install:  false
  imageTag:  v1.16.bbbb
  imageRep:  acs/storage-analyzer
  template:  /acs/templates/storage-analyzer/install.yaml
  # env config
  LOOP_INTERVAL: 30
 storage-auto-expander: |
  # deploy config
  install:  false
  imageTag:  v1.16.3adsadfs
  imageRep:  acs/storage-auto-expander
  crdTmpl:  /acs/templates/storage-auto-expander/crd.yaml
  template:  /acs/templates/storage-auto-expander/install.yaml
  # env config
  POLLING_INTERVAL_SECONDS: 60
```

Take note of the following items when you use custom configurations:

- The configurations of each component consist of two parts:
  - Deployment configurations, which are defined by the parameters in the deploy config section in the preceding example. The parameters specify whether to install the component and the component version that is deployed. Storage Operator deploys and upgrades components based on these configurations.
  - Environment configurations, which are defined by the parameters in the env config section in the preceding example. The parameters are used by the storage component. For example, `SNAPSHOT_INTERVAL` is configured as an environment variable for the storage-snapshot-manager component.
- The parameter values that are specified in the custom configurations overwrite the parameter values in the default configurations.
- The parameters used in deployment configurations are reserved parameters and cannot be used as environment variables.
- You can customize specific parameters based on the features of the component.

Step 2: Deploy Storage Operator

1. Run the following command to deploy Storage Operator:

```
kubectl apply -f StorageOperator.yaml
```

In this example, the following *StorageOperator.yaml* file is used to deploy Storage Operator:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: storage-operator
 namespace: kube-system
 labels:
  app: storage-operator
spec:
 selector:
  matchLabels:
   app: storage-operator
 template:
  metadata:
   labels:
    app: storage-operator
  spec:
   tolerations:
    - operator: "Exists"
   priorityClassName: system-node-critical
   serviceAccount: storage-operator-admin
   containers:
   - name: storage-operator
    image: registry.cn-hangzhou.aliyuncs.com/acs/storage-operator:v1.18.8.28-18cca7b-aliyun
    imagePullPolicy: Always
    volumeMounts:
    - mountPath: /acs/configmap/
      name: storage-operator
   restartPolicy: Always
   volumes:
   - configMap:
     name: storage-operator
    name: storage-operator
```

2. Run the following command to check whether Storage Operator is running:

```
kubectl get pods -nkubes-system | grep storage-operator
```

Expected output:

```
NAME              READY  STATUS   RESTARTS  AGE    IP          NODE          NOMINATED NODE  READINESS G
ATES
storage-operator-***    1/1   Running  0     6d20h   192.168.XX.XX  virtual-kubelet  <none>      <none>
```

8.11. FAQ about CSI

This topic describes how to troubleshoot common issues related to storage and how to resolve common issue related to disk volumes and NAS volumes.

Troubleshoot common issues

Perform the following operations to view the log of a specified volume plug-in. This allows you to identify the problems.

1. Run the following command to check whether events occur in persistent volume claims (PVCs) or pods:

```
kubectl get events
```

Expected output:

```
LAST SEEN  TYPE    REASON          OBJECT                                  MESSAGE
2m56s      Normal  FailedBinding   persistentvolumeclaim/data-my-release-mariadb-0      no persistent volumes available for this claim and no storage class is set
41s        Normal  ExternalProvisioning  persistentvolumeclaim/pvc-nas-dynamic-create-subpath8  waiting for a volume to be created, either by external provisioner "nasplugin.csi.alibabacloud.com" or manually created by system administrator
3m31s      Normal  Provisioning    persistentvolumeclaim/pvc-nas-dynamic-create-subpath8  External provisioner is provisioning volume for claim "default/pvc-nas-dynamic-create-subpath8"
```

2. Run the following command to check whether the FlexVolume or CSI plug-in is deployed in the cluster:
   - Run the following command to check whether the FlexVolume plug-in is deployed in the cluster:

     ```
     kubectl get pod -nkube-system |grep flexvolume
     ```

     Expected output:

     ```
     NAME              READY  STATUS       RESTARTS  AGE
     flexvolume-***       4/4    Running      0      23d
     ```

   - Run the following command to check whether the CSI plug-in is deployed in the cluster:

     ```
     kubectl get pod -nkube-system |grep csi
     ```

     Expected output:

     ```
     NAME              READY  STATUS       RESTARTS  AGE
     csi-plugin-***       4/4    Running      0      23d
     csi-provisioner-***     7/7    Running      0      14d
     ```

3. Check whether the volume templates match the template of the volume plug-in used in the cluster. The supported volume plug-ins are FlexVolume and CSI.

   If this is the first time you mount volumes in the cluster, check the driver specified in the persistent volume (PV) and StorageClass. The driver that you specified must be the same as the volume plug-in that is deployed in the cluster.

4. Check whether the volume plug-in is upgraded to the latest version.
   - Run the following command to query the image version of the FlexVolume plug-in:

     ```
     kubectl get ds flexvolume -nkube-system -oyaml | grep image
     ```

     Expected output:

     ```
     image: registry.cn-hangzhou.aliyuncs.com/acs/Flexvolume:v1.14.8.109-649dc5a-aliyun
     ```

     For more information about the FlexVolume plug-in, see Flexvolume.

   - Run the following command to query the image version of the CSI plug-in:

     ```
     kubectl get ds csi-plugin -nkube-system -oyaml |grep image
     ```

     Expected output:

     ```
     image: registry.cn-hangzhou.aliyuncs.com/acs/csi-plugin:v1.18.8.45-1c5d2cd1-aliyun
     ```

     For more information about the CSI plug-in, see csi-plugin and csi-provisioner.

5. View log data.
   - If a PVC of disk type is in the Pending state, the cluster fails to create the PV. You must check the log

of the Provisioner plug-in.

- If the FlexVolume plug-in is deployed in the cluster, run the following command to print the log of alicloud-disk-controller:

```
podid=`kubectl get pod -nkube-system | grep alicloud-disk-controller | awk '{print $1}'`
kubectl logs {PodID} -nkube-system
```

- If the CSI plug-in is deployed in the cluster, run the following command to print the log of csi-provisioner:

```
podid=`kubectl get pod -nkube-system | grep csi-provisioner | awk '{print $1}'`
kubectl logs {PodID} -nkube-system -c csi-provisioner
```

> **Note** Two pods are created to run csi-provisioner. After you run the `kubectl get pod -nkube-system | grep csi-provisioner | awk '{print $1}'` command, two `podid` are returned. Then, run the `kubectl logs {PodID} -nkube-system -c csi-provisioner` command for each of the two pods.

- If a mounting error occurs when the system starts a pod, you must check the log of FlexVolume or csi-plugin.
  - If the FlexVolume plug-in is deployed in the cluster, run the following command to print the log of FlexVolume:

```
kubectl get pod {pod-name} -owide
```

    Log on to the Elastic Compute Service (ECS) instance where the pod runs and check the log of FlexVolume in the `/var/log/alicloud/flexvolume_**.log` directory.
  - If the CSI plug-in is deployed in the cluster, run the following command to print the log of csi-plugin:

```
nodeID=`kubectl get pod {pod-name} -owide | awk 'NR>1 {print $7}'`
podID=`kubectl get pods -nkube-system -owide -lapp=csi-plugin | grep $nodeID|awk '{print $1}'`
kubectl logs {PodID} -nkube-system
```

- View the log of Kubelet.
  Run the following command to query the node where the pod runs:

```
kubectl get pod deployment-disk-5c795d7976-bjhkj -owide | awk 'NR>1 {print $7}'
```

  Log on to the node and check the log in the */var/log/message* directory.

Quick recovery

If you fail to mount volumes to most of the pods on a node, you can schedule the pods to another node. For more information, see the questions and answers in the following sections.

FAQ about disk volumes

does the system prompt "disk size is not supported" when I dynamically provision a

?❯

generates a pod log that is not managed by Container Service for Kubernetes (ACK)?❯

FAQ about NAS volumes

the system prompt "chown: option not permitted" when I mount a NAS file system?❯

9.Storage management-Flexvolume

9.1. Overview

Container Service for Kubernetes (ACK) clusters can be automatically bound to disks of Alibaba Cloud, Aspara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets that are mounted to pods. This topic describes the storage services that are supported by ACK and how to use these services. ACK supports static volumes and dynamic volumes.

Alibaba Cloud storage	Static volume	Dynamic volume
Alibaba Cloud disk	You can statically provision a disk volume in the following ways: • Directly create a disk volume • Create a pair of persistent volume (PV) and persistent volume claim (PVC)	Supported

NAS file system	You can statically provision a NAS volume in the following ways:
	• Use the FlexVolume plug-in
	  ◦ Directly create a NAS volume
	  ◦ Create a pair of persistent volume (PV) and persistent volume claim (PVC)
	• Use the Network File System (NFS) driver
		Supported

OSS bucket	You can statically provision an OSS volume in the following ways: • Directly create a disk volume • Create a pair of persistent volume (PV) and persistent volume claim (PVC)	Not supported

9.2. Volume plug-ins

Clusters of Container Service for Kubernetes (ACK) support the Flexvolume and CSI plug-ins. This topic describes the features of these plug-ins and how to select between them based on your requirements.

Differences between FlexVolume and CSI

Plug-in	Feature	Related topic

Plug-in	Feature	Related topic
FlexVolume	FlexVolume is a traditional mechanism to extend Kubernetes storage systems developed by the Kubernetes community. ACK supports FlexVolume. FlexVolume consists of the following parts: • FlexVolume: Allows you to mount and delete volumes and PVs. By default, ACK allows you to mount the following types of storage media: disks, Network Storage Service (NAS) file systems, and Object Storage Service (OSS) buckets. • Disk-Controller: Automatically creates disks. • Nas-Controller: Automatically creates NAS file systems.	For more information about FlexVolume, see Overview. For more information about upgrading FlexVolume, see Manage system components.
CSI	CSI is recommended by the Kubernetes community. The CSI plug-in provided by ACK is compatible with the features of its community version. CSI consists of the following parts: • CSI-Plugin: Allows you to mount and delete volumes and PVs. By default, ACK allows you to mount the following types of storage media: disks, NAS file systems, and OSS buckets. • CSI-Provisioner: Automatically creates volumes, PVs, and related disks or NAS file systems.	For more information about CSI, see Overview and alibaba-cloud-csi-driver.

Recommendations

- For newly created ACK clusters, we recommend that you use CSI. The ACK technical team will continue to upgrade CSI to support more features of its community version.
- For existing ACK clusters, we recommend that you use the preinstalled plug-in. The ACK technical team will continue to support FlexVolume.

Considerations

- You need to select a plug-in when you create an ACK cluster.
- You cannot use both plug-ins on the same ACK cluster.
- You cannot change the plug-in from FlexVolume to CSI for an ACK cluster.

9.3. Install and upgrade FlexVolume

If you specify FlexVolume as the volume plug-in for a cluster of Container Service for Kubernetes (ACK) earlier than 1.16, the system installs FlexVolume and Disk Controller in the cluster by default. However, the system does not automatically install NAS Controller. This topic describes how to install and upgrade FlexVolume and how to install NAS Controller.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- FlexVolume is specified as the volume plug-in of the ACK cluster.
- You are connected to the cluster by using kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

Limits

Only CentOS 7 and Aliyun Linux 2 are supported.

Install the components

Install FlexVolume

- Clusters of ACK 1.16 and later do not support FlexVolume. You must install CSI-Plugin in these clusters. For more information, see Differences between the CSI and FlexVolume plug-ins.
- If you specify FlexVolume as the volume plug-in for a cluster of ACK earlier than 1.16, the system installs FlexVolume in the cluster by default. For more information, see Component configurations.

Install Disk Controller

- Clusters of ACK 1.16 and later do not support Disk Controller. You must install CSI-Provisioner in these clusters. For more information, see Differences between the CSI and FlexVolume plug-ins.
- If you specify FlexVolume as the volume plug-in for a cluster of ACK earlier than 1.16, the system installs Disk Controller in the cluster by default. For more information, see Component configurations.

Install NAS Controller

If your cluster already has FlexVolume installed, you can manually install NAS Controller and then dynamically provision volumes that use Apsara File Storage (NAS) file systems.

You can use the following YAML template to manually install NAS Controller:

```
kind: Deployment
apiVersion: apps/v1
metadata:
 name: alicloud-nas-controller
 namespace: kube-system
spec:
 selector:
  matchLabels:
   app: alicloud-nas-controller
 strategy:
  type: Recreate
 template:
  metadata:
   labels:
    app: alicloud-nas-controller
  spec:
   tolerations:
   - operator: Exists
   affinity:
    nodeAffinity:
     preferredDuringSchedulingIgnoredDuringExecution:
     - weight: 1
       preference:
        matchExpressions:
        - key: node-role.kubernetes.io/master
          operator: Exists
   priorityClassName: system-node-critical
   serviceAccount: admin
   hostNetwork: true
   containers:
    - name: nfs-provisioner
      image: registry.cn-hangzhou.aliyuncs.com/acs/alicloud-nas-controller:v1.14.3.8-58bf821-aliyun
      env:
      - name: PROVISIONER_NAME
        value: alicloud/nas
      securityContext:
        privileged: true
      volumeMounts:
      - mountPath: /var/log
        name: log
   volumes:
   - hostPath:
     path: /var/log
    name: log
```

Verify the installation

Check whether FlexVolume, Disk Controller, and NAS Controller are installed in the cluster.

- Run the following command to check whether FlexVolume is installed in the cluster:

  ```
  kubectl get pod -nkube-system | grep flexvolume
  ```

- Run the following command to check whether Disk Controller is installed in the cluster:

  ```
  kubectl get pod -nkube-system | grep alicloud-disk-controller
  ```

- Run the following command to check whether NAS Controller is installed in the cluster:

```
kubectl get pod -nkube-system | grep alicloud-nas-controller
```

Upgrade the components

You can upgrade FlexVolume and Disk Controller in the ACK console. You cannot upgrade NAS Controller in the ACK console.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. Click the **Storage** tab, find **flexvolume** and **alicloud-disk-controller**, and click **Upgrade**.
6. In the **Note** message, confirm the versions of the plug-ins and click **OK**.
   After the plug-ins are upgraded, the system prompts that the upgrades are completed and the current versions of the plug-ins are displayed.

- When you upgrade FlexVolume in the following scenarios, Submit a ticket to request technical support.
  - The system fails to update FlexVolume in the ACK console.
  - The version of FlexVolume is 1.12 or earlier, and volumes that use disks and Object Storage Service (OSS) buckets are provisioned in the cluster.
  - You want to ensure a successful upgrade because sensitive business data is stored in the cluster and a large number of volumes are used.
- The system fails to upgrade Disk Controller. In this case, Submit a ticket to request technical support.

9.4. Disk volumes

9.4.1. Usage notes for disk volumes

This topic describes how to use Alibaba Cloud disks that are mounted as volumes in Container Service for Kubernetes (ACK) clusters.

Background information

Alibaba Cloud disks can be mounted to ACK clusters in the following types:

- Statically provisioned volumes
  You can use statically provisioned disk volumes in the following ways:
  - Use disks as volumes.
  - Use disks by creating persistent volumes (PVs) and persistent volume claims (PVCs).
- Dynamically provisioned volumes

Usage notes

- A disk cannot be shared. If you use a disk by creating a pair of PV and PVC, the disk can be mounted to only one pod.
- We recommend that you use a StatefulSet to mount a disk. To mount a disk by using a Deployment, you must set the number of replicated pods to 1. If you use a Deployment to mount a disk, you must set the number of replicated pods to 1. If the number of replicated pods is not set to 1, multiple nodes may share the disk when the pods are scheduled to different nodes. In this case, you cannot prioritize the node where you want to mount or unmount a disk. We recommend that you do not use this method to mount

disks.

- Before you can use a statically provisioned disk volume, you must create the disk and obtain the disk ID. For more information, see Create a disk.

  > **Note** Your disk must meet the following requirements:
  > - An ultra disk must have a minimum capacity of 20 GiB.
  > - A standard SSD must have a minimum capacity of 20 GiB.
  > - An enhanced SSD (ESSD) must have a minimum capacity of 20 GiB.

- VolumeId: the ID of the disk that you want to mount. The value must be the same as the value of VolumeName and PV Name.
- You can mount disks to only the nodes that are deployed in the same zone as the disks. Therefore, when you create a disk, select the zone of the pod to which you want to mount the disk.
- Only pay-as-you-go disks can be mounted. If you change the billing method of an Elastic Compute Service (ECS) instance in the cluster from pay-as-you-go to subscription, you cannot change the billing method of its disks to subscription. Otherwise, the disks cannot be mounted to the cluster.
- Some ECS instances do not support ESSDs. If this type of ECS instance is used as a node, you cannot mount an ESSD to the node. For more information, see Elastic Block Storage FAQ.
- If a disk is partitioned, you must re-initialize the disk before you mount it as a volume.

9.4.2. Use Alibaba Cloud disks as statically provisioned volumes

You can use Alibaba Cloud disks as volumes. You can also use disks to create persistent volumes (PVs) or persistent volume claims (PVCs).

Prerequisites

To use a disk as a volume, you must first create the disk in the Elastic Compute Service (ECS) console. For more information, see Create a disk.

Use a disk as a volume

Use the following *disk-deploy.yaml* file to create a pod.

1. Create the *disk-deploy.yaml* file and copy the following content to the file.

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 labels:
  app: nginx
spec:
 ports:
 - port: 80
  name: web
 clusterIP: None
 selector:
  app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: web
spec:
 selector:
  matchLabels:
   app: nginx
 serviceName: "nginx"
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
     name: web
    volumeMounts:
    - name: d-wz9f6l6nm0uvazgh7y17
     mountPath: /data
   volumes:
    - name: "d-wz9f6l6nm0uvazgh7y17"
     flexVolume:
      driver: "alicloud/disk"
      fsType: "ext4"
      options:
       volumeId: "d-wz9f6l6nm0uvazgh7y17"
```

2. Run the following command to create a pod:

```
kubectl apply -f disk-deploy.yaml
```

> **Note**
> In the `volumes` section, we recommend that you set `name` and `volumeId` to the same value.
> If your cluster is deployed across multiple zones, schedule the pod to the zone where the disk is provisioned. Run the following command to add a nodeSelector field to the pod:

```
nodeSelector:
    failure-domain.beta.kubernetes.io/zone: cn-hangzhou-b
```

Use disks to create PVs or PVCs

1. Use a disk to create a PV.

   You can create a PV in the Container Service for Kubernetes (ACK) console or by using a YAML file.

   - Create a PV by using a YAML file
     Use the following *disk-pv.yaml* file to create a PV.

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: d-bp1j17ifxfasvts3****
 labels:
  failure-domain.beta.kubernetes.io/zone: cn-hangzhou-b
  failure-domain.beta.kubernetes.io/region: cn-hangzhou
spec:
 capacity:
  storage: 20Gi
 storageClassName: disk
 accessModes:
  - ReadWriteOnce
 flexVolume:
  driver: "alicloud/disk"
  fsType: "ext4"
  options:
   volumeId: "d-bp1j17ifxfasvts3****"
```

     > **Note** The `name` field value of the PV must be the same as the disk ID that is specified by `volumeId`.

   - Create a PV in the ACK console
     a. Log on to the ACK console.
     b. In the left-side navigation pane, click **Clusters**.
     c. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column of the cluster.
     d. The Cluster Information page appears. In the left-side navigation pane, click **Persistent Volumes**.
     e. On the Persistent Volumes page, click the **Persistent Volumes** tab and click **Create** in the upper-right corner of the page.

f. Set the parameters in the **Create PV** dialog box.

Parameter	Description
PV Type	In this example, Cloud Disk is selected.
Volume Plug-in	In this example, **Flexvolume** is selected.
Access Mode	By default, ReadWriteOnce is used.
Disk ID	Select a disk that is in the same region and zone as your cluster.
File System Type	Select the file system of the disk. Supported file systems include ext4, ext3, xfs, and vfat. Default value: ext4.
Label	Add labels to the PV.

g. After you complete the settings, click **Create**.

2. Create a PVC.

   Use the following *disk-pvc.yaml* file to create a PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: pvc-disk
spec:
 accessModes:
  - ReadWriteOnce
 storageClassName: disk
 resources:
  requests:
   storage: 20Gi
```

3. Create a pod.

   Use the following disk-pod.yaml file to create a pod:

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 labels:
  app: nginx
spec:
 ports:
 - port: 80
  name: web
 clusterIP: None
 selector:
  app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: web
spec:
 selector:
  matchLabels:
   app: nginx
 serviceName: "nginx"
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
     name: web
    volumeMounts:
    - name: pvc-disk
     mountPath: /data
   volumes:
    - name: pvc-disk
     persistentVolumeClaim:
      claimName: pvc-disk
```

9.4.3. Dynamically provision a disk volume by using the CLI

To dynamically provision a disk as a persistent volume (PV), you must manually create a StorageClass, and set the storageClassName field in a persistent volume claim (PVC) to specify the disk type.

Create a StorageClass with a specified zone ID (zoneId)

1. Create a *storage-class.yaml* file and copy the following content to the file:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: alicloud-disk-ssd-hangzhou-b
provisioner: alicloud/disk
parameters:
 type: cloud_ssd
 regionId: cn-hangzhou
 zoneId: cn-hangzhou-b
reclaimPolicy: Retain
```

The following table describes the parameters.

Parameter	Description
provisioner	Set the value to alicloud/disk. This is a volume plug-in used to provision disks.
type	The disk type. Valid values include cloud_efficiency, cloud_ssd, cloud_essd, and available. If you set this parameter to available, the system attempts to create a disk in the following order: enhanced SSD (ESSD), standard SSD, and ultra disk. The system keeps trying until a disk is created.
regionId	The region where you want to create the disk.
reclaimPolicy	The policy to reclaim the disk. By default, this parameter is set to Delete. You can also set this parameter to Retain. If you require high data security, we recommend that you set this parameter to Retain to avoid data loss caused by user errors.
zoneId	The zone where you want to create the disk. For a multi-zone cluster, you can specify multiple zones. Example: `zoneId: cn-hangzhou-a,cn-hangzhou-b,cn-hangzhou-c`
encrypted	Optional. This parameter specifies whether the disk is encrypted. By default, this parameter is set to false. This specifies that the disk is not encrypted.

2. Run the following command to create a StorageClass:

```
kubectl apply -f storage-class.yaml
```

Create a StorageClass in WaitForFirstConsumer mode

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: alicloud-disk-topology-ssd
provisioner: alicloud/disk
parameters:
   type: cloud_ssd
reclaimPolicy: Retain
volumeBindingMode: WaitForFirstConsumer
```

> **Note**
>
> - If you do not create a StorageClass in WaitForFirstConsumer mode and the zoneid parameter is not set, a PV is created in the zone where the Disk-Controller component is deployed.
> - If you do not create a StorageClass in WaitForFirstConsumer mode but the zoneid parameter is set, the system attempts to create a PV in the specified zones based on the round-robin algorithm.
> - If you create a StorageClass in WaitForFirstConsumer mode, a disk is created for the node to which the pod that consumes the PVC is scheduled. The disk is created in the zone of the scheduled pod.

Create a PVC

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: disk-ssd
spec:
 accessModes:
  - ReadWriteOnce
 storageClassName: alicloud-disk-ssd-hangzhou-b
 resources:
  requests:
   storage: 20Gi
---
kind: Pod
apiVersion: v1
metadata:
 name: disk-pod-ssd
spec:
 containers:
 - name: disk-pod
  image: nginx
  volumeMounts:
   - name: disk-pvc
     mountPath: "/mnt"
 restartPolicy: "Never"
 volumes:
  - name: disk-pvc
    persistentVolumeClaim:
     claimName: disk-ssd
```

The following default settings are included:

In a multi-zone cluster, you must manually create a StorageClass to specify the zone where a disk is created.

By default, the following types of StorageClass are provided for single-zone clusters:

- alicloud-disk-efficiency: ultra disk.
- alicloud-disk-ssd: standard SSD.
- alicloud-disk-essd: ESSD.
- alicloud-disk-available: a high-availability mode. In this mode, the system first attempts to create a standard SSD. If SSD resources are exhausted, the system attempts to create an ultra disk.

> **Notice** For alicloud-disk-controller versions earlier than v1.14.8.44-c23b62c5-aliyun, the system attempts to create a disk in the following order: ESSD, standard SSD, and ultra disk. The system keeps trying until a disk is created.

- alicloud-disk-topology: creates a disk in WaitForFirstConsumer mode.

Create a multi-instance StatefulSet by using a disk

You can use the volumeClaimTemplates parameter to dynamically create multiple PVCs and PVs and bind the PVs to PVCs.

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 labels:
  app: nginx
spec:
 ports:
 - port: 80
  name: web
 clusterIP: None
 selector:
  app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: web
spec:
 selector:
  matchLabels:
   app: nginx
 serviceName: "nginx"
 replicas: 2
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
     name: web
    volumeMounts:
    - name: disk-ssd
     mountPath: /data
 volumeClaimTemplates:
 - metadata:
   name: disk-ssd
  spec:
   accessModes: [ "ReadWriteOnce" ]
   storageClassName: "alicloud-disk-ssd-hangzhou-b"
   resources:
    requests:
     storage: 20Gi
```

You can also use dynamically provisioned disk volumes in the Container Service for Kubernetes (ACK) console. For more information, see Use a dynamically provisioned disk volume in the ACK console.

9.4.4. Use a dynamically provisioned disk volume in the ACK console

This topic describes how to use a dynamically provisioned disk volume in the Container Service for Kubernetes (ACK) console.

Prerequisites

An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.

Create a StorageClass

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Volumes > StorageClasses**.
5. On the **StorageClasses** page, click **Create**. In the **Create** dialog box that appears, set the following parameters:
   - **Name**: the name of the disk.
   - **PV Type**: Set the value to Cloud Disk. This specifies that the provisioner plug-in for Alibaba Cloud disks is used to create the StorageClass.
   - **Volume Plug-in**: In this example, **Flexvolume** is selected.
   - **Parameter**: In this example, the parameters are type and zoneid.
     - type: the disk type. Valid values: cloud_efficiency, cloud_ssd, cloud_essd, and available. If you set this parameter to available, the system attempts to create a disk until it succeeds. The system selects a disk type in sequence from this list: an enhanced SSD (ESSD), a standard SSD, and an ultra disk. The system keeps trying until a disk is created.
     - zoneid: This parameter specifies the region where the disk is created.
       For a multi-zone cluster, you can specify multiple zones. Example:

       ```
       zoneid: cn-hangzhou-a,cn-hangzhou-b,cn-hangzhou-c
       ```

     - encrypted: optional. This parameter specifies whether the disk to be created is encrypted. By default, this parameter is set to false. This specifies that the disk to be created is not encrypted.
   - **Reclaim Policy**: the policy that is used to reclaim a disk. By default, this parameter is set to Delete. You can also set this parameter to Retain. If you require higher data security, we recommend that you set this parameter to Retain to avoid data loss caused by user errors.
   - **Binding Mode**: Valid values: Immediate and WaitForFirstConsumer. Default value: Immediate.
   - **Mount Options**: When you mount a volume, you can add multiple mount options.
6. Click **Create**.

Create a PVC

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volume Claims**.
5. In the upper-right corner of the **Persistent Volume Claims** page, click **Create**. In the **Create PVC** dialog box, set the required parameters.
   - **PVC Type**: **Cloud Disk**, **NAS**, and **OSS** are supported. In this example, **Cloud Disk** is selected.

- **Name**: the name of the persistent volume claim (PVC). The name must be unique in the namespace.
- **Allocation Mode**: **Use StorageClass**, **Existing Volumes**, and **Create Volume** are supported. In this example, **Use StorageClass** is selected.
- **Existing Storage Class**: Click **Select**. Find the StorageClass that you want to use and click **Select** in the Actions column.
- **Capacity**: the capacity of the PVC.

  > **Note** The capacity of the PVC cannot exceed the capacity of the disk.

- **Access Mode**: The default value is ReadWriteOnce.

6. Click **Create**.
   After the PVC is created, the PVC named test-cloud appears in the list of PVCs. The PVC is associated with the specified PV.

Use the dynamically provisioned disk volume

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from Image**.
6. Set the parameters that are required to create a Deployment.

   This example shows how to set the volume parameters. For more information about other parameters, see Create a stateless application by using a Deployment.
   You can add local volumes and cloud volumes.

   - **Local Storage**: You can select HostPath, ConfigMap, Secret, or EmptyDir. The source directory or file is mounted to a path in the container. For more information, see Volumes.
   - **Cloud Storage**: supports the following types of persistent volumes (PVs): disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS).

   In this example, a PV is created from a disk, and the PV is mounted to the */tmp* path in the container. The data that is generated in this path is stored in the disk.
7. Set other parameters and click **Create**.

   After the disk volume is created, you can use the disk volume.

You can also run commands to create a dynamically provisioned disk volume. For more information, see Dynamically provision a disk volume by using the CLI.

9.4.5. Use statically provisioned disk volumes for persistent storage

When a node that hosts running containers fails, stateful applications may lose the business data stored in the containers. This issue can be resolved by using persistent storage. This topic describes how to use a statically provisioned disk volume to persist data.

Prerequisites

Before you start, make sure that you have performed the following operations:

- 创建Kubernetes托管版集群

- Create a disk
- Connect to Kubernetes clusters by using kubectl

Context

Scenarios

- Provide storage space for applications that require high disk I/O and do not require data sharing. For example, storage services such as MySQL and Redis.
- Write log data at high speeds.
- Store data in a way that is independent of the lifetime of a pod.

You must create a disk before you can statically provision a disk volume.

You must manually create a persistent volume (PV) and a persistent volume claim (PVC) to use a statically provisioned disk volume. For more information, see Use Alibaba Cloud disks as statically provisioned volumes.

Limits

- The disks provided by Alibaba Cloud cannot be shared. Each disk can be mounted only to one pod.
- A disk can be mounted only to a node that is deployed in the same zone as the disk.

Create a PV

1. Create a file named *pv-static.yaml*.

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: <your-disk-id>
 labels:
  alicloud-pvname: <your-disk-id>
  failure-domain.beta.kubernetes.io/zone: <your-zone>
  failure-domain.beta.kubernetes.io/region: <your-region>
spec:
 capacity:
  storage: 20Gi
 accessModes:
  - ReadWriteOnce
 flexVolume:
  driver: "alicloud/disk"
  fsType: "ext4"
  options:
   volumeId: "<your-disk-id>"
```

> **Note**
>
> - `alicloud-pvname: <your-disk-id>`: the name of the PV. Set the value to the disk ID.
> - `failure-domain.beta.kubernetes.io/zone: <your-zone>`: the zone where the disk is deployed. Example: *cn-hangzhou-b*.
> - `failure-domain.beta.kubernetes.io/region: <your-region>`: the region where the disk is deployed. Example: *cn-hangzhou*.
>
> If your cluster is deployed across zones, you must specify `failure-domain.beta.kubernetes.io/zone` and `failure-domain.beta.kubernetes.io/region`. This ensures that your pods are scheduled to the zone where the disk is deployed.

2. Run the following command to create a PV:

```
kubectl create -f pv-static.yaml
```

Verify the result

i. Log on to the ACK console.

ii. In the left-side navigation pane, click **Clusters**.

iii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.

iv. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**. Verify that the newly created PV is displayed.

Create a PVC

1. Create a file named *pvc-static.yaml*.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: pvc-disk
spec:
 accessModes:
  - ReadWriteOnce
 resources:
  requests:
   storage: 20Gi
 selector:
  matchLabels:
   alicloud-pvname: <your-disk-id>
```

2. Run the following command to create a PVC:

```
kubectl create -f pvc-static.yaml
```

View the result

i. Log on to the ACK console.

ii. In the left-side navigation pane, click **Clusters**.

iii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.

iv. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volume Claims**.

v. On the **Persistent Volume Claims** page, verify that the newly created PVC is displayed.

Create an application

1. Create a file named *static.yaml*.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-static
 labels:
  app: nginx
spec:
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    volumeMounts:
     - name: disk-pvc
      mountPath: "/data"
   volumes:
    - name: disk-pvc
     persistentVolumeClaim:
      claimName: pvc-disk
```

2. Run the following command to deploy an application that uses the statically provisioned disk volume:

```
kubectl create -f static.yaml
```

View the result

i. Log on to the ACK console.

ii. In the left-side navigation pane, click **Clusters**.

iii. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.

iv. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

v. On the **Deployments** page, verify that the newly created application is displayed.

Verify data persistence

1. Run the following command to query the pods that host the application:

```
kubectl get pod | grep static
```

Expected output:

```
nginx-static-78c7dcb9d7-g****  2/2   Running   0     32s
```

2. Run the following command to check whether a new disk is mounted to the */data* path:

```
kubectl exec nginx-static-78c7dcb9d7-g**** df | grep data
```

Expected output:

```
/dev/vdf     20511312   45080 20449848   1% /data
```

3. Run the following command to query the files in the */data* path:

```
kubectl exec nginx-static-78c7dcb9d7-g**** ls /data
```

Expected output:

```
lost+found
```

4. Run the following command to create a file named *static* in the */data* path:

```
kubectl exec nginx-static-78c7dcb9d7-g**** touch /data/static
```

5. Run the following command to query the files in the */data* path:

```
kubectl exec nginx-static-78c7dcb9d7-g**** ls /data
```

Expected output:

```
static
lost+found
```

6. Run the following command to delete the pod named `nginx-static-78c7dcb9d7-g****`.

```
kubectl delete pod nginx-static-78c7dcb9d7-g****
```

Expected output:

```
pod "nginx-static-78c7dcb9d7-g****" deleted
```

7. Open another kubectl command-line interface (CLI) and run the following command to view how the pod is deleted and recreated:

```
kubectl get pod -w -l app=nginx
```

Expected output:

```
NAME                     READY  STATUS      RESTARTS  AGE
nginx-static-78c7dcb9d7-g****  2/2   Running      0     50s
nginx-static-78c7dcb9d7-g****  2/2   Terminating    0     72s
nginx-static-78c7dcb9d7-h****  0/2   Pending      0     0s
nginx-static-78c7dcb9d7-h****  0/2   Pending      0     0s
nginx-static-78c7dcb9d7-h****  0/2   Init:0/1     0     0s
nginx-static-78c7dcb9d7-g****  0/2   Terminating    0     73s
nginx-static-78c7dcb9d7-h****  0/2   Init:0/1     0     5s
nginx-static-78c7dcb9d7-g****  0/2   Terminating    0     78s
nginx-static-78c7dcb9d7-g****  0/2   Terminating    0     78s
nginx-static-78c7dcb9d7-h****  0/2   PodInitializing  0     6s
nginx-static-78c7dcb9d7-h****  2/2   Running      0     8s
```

8. Run the following command to query the recreated pod:

```
kubectl get pod
```

Expected output:

```
NAME                     READY  STATUS   RESTARTS  AGE
nginx-static-78c7dcb9d7-h****  2/2   Running  0     14s
```

9. Run the following command to verify that the *static* file still exists in the */data* path. This indicates that data is persisted to the disk.

```
kubectl exec nginx-static-78c7dcb9d7-h6brd ls /data
```

Expected output:

```
static
lost+found
```

9.4.6. Use dynamically provisioned disks for stateful applications

This topic describes when and how to use dynamically provisioned disks for stateful applications.

Background information

When to use a dynamically provisioned disk:

If no disk is available when you deploy an application in a Container Service for Kubernetes (ACK) cluster, the system automatically creates a disk and uses it as a dynamic volume.

How to use a dynamically provisioned disk:

1. Manually create a persistent volume claim (PVC) and specify a StorageClass in the PVC.
2. When you deploy an application, the system automatically creates a persistent volume (PV) of the specified StorageClass.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- You are connected to the cluster through kubectl. For more information, see Connect to an ACK cluster by using kubectl.
- A provisioner is installed for the cluster. The provisioner can automatically create a disk of the specified StorageClass.

Provisioner

By default, a provisioner is automatically installed in each ACK cluster.

Create a StorageClass

By default, ACK creates four StorageClasses when the system is initialized. All of the StorageClasses use the default settings. Only single-zone clusters can use the four StorageClasses to dynamically provision disks. If your cluster is deployed in more than one zone, you must create new StorageClasses to mount disks as dynamic volumes. The following list describes the four StorageClasses:

- *alicloud-disk-common*: creates a basic disk.
- *alicloud-disk-efficiency*: creates an ultra disk.
- *alicloud-disk-ssd*: creates an SSD.
- *alicloud-disk-available*: a high-availability mode. In this mode, the system attempts to create an SSD. If SSD resources are exhausted, the system then attempts to create an ultra disk.

1. Create a *storageclass.yaml* file.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: alicloud-disk-ssd-hangzhou-b
provisioner: alicloud/disk
reclaimPolicy: Retain
parameters:
 type: cloud_ssd
 regionId: cn-hangzhou
 zoneId: cn-hangzhou-b
 fstype: "ext4"
 readonly: "false"
```

Parameter	Description
provisioner	The value is set to alicloud/disk to automatically create a disk by using the provisioner.
reclaimPolicy	Specifies how the disk is handled after the corresponding PVC is deleted. Valid values: *Delete* and *Retain*. Default value: *Delete*. **Note** If you set the value to *Delete*, the disk is automatically deleted when you delete the PVC. The disk data cannot be restored.
type	The type of disk. Valid values: *cloud*, *cloud_efficiency*, *cloud_ssd*, and *available*.
regionId	The ID of the region to which the disk belongs. The value must be set to the region ID of the cluster. This parameter is optional.
zoneId	The ID of the zone to which the disk belongs. This parameter is optional. ◦ If your cluster is deployed in one zone, set the value to the ID of the zone. ◦ If your cluster is deployed in multiple zones, set the value to the IDs of the zones. Example: `zoneid: cn-hangzhou-a,cn-hangzhou-b,cn-hangzhou-c`
fstype	The file system of the disk. This parameter is optional. Default value: *ext4*.
readonly	Specifies whether the disk is read-only. This parameter is optional. Valid values: true and false. *true*: The disk is read-only. *false*: You can perform read and write operations on the disk. Default value: *false*.
encrypted	Specifies whether to encrypt the disk. This parameter is optional. Valid values: true and false. *true*: encrypts the disk. *false*: does not encrypt the disk. Default value: *false*.

2. Run the following command to create a StorageClass:

```
kubectl create -f storageclass.yaml
```

Create a PVC

1. Create a *pvc-ssd.yaml* file.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: disk-ssd
spec:
 accessModes:
  - ReadWriteOnce
 storageClassName: alicloud-disk-ssd-hangzhou-b
 resources:
  requests:
   storage: 20Gi
```

2. Run the following command to create a PVC:

```
kubectl create -f pvc-ssd.yaml
```

Expected results

On the details page of the **cluster**, click **Volumes** and choose **Persistent Volume Claims** in the left-side navigation pane. On the **Persistent Volume Claims** page, you can view the created PVC and the PV that is bound to the PVC. The `StorageClass` of the PVC is *alicloud-disk-ssd-hangzhou-b*.

Create a Deployment

1. Create a *pvc-dynamic.yaml* file.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-dynamic
 labels:
  app: nginx
spec:
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    volumeMounts:
     - name: disk-pvc
       mountPath: "/data"
   volumes:
    - name: disk-pvc
      persistentVolumeClaim:
       claimName: disk-ssd
```

2. Run the following command to create a Deployment:

```
kubectl create -f nginx-dynamic.yaml
```

Expected results

On the details page of the **cluster**, click **Workloads** and choose **Deployments** in the left-side navigation pane. On the **Deployments** page, you can view the created Deployment.

Use the disk for persistent storage

1. Run the following command to query the pod that hosts the Deployment:

   ```
   kubectl get pod | grep dynamic
   ```

   The following output is returned:

   ```
   nginx-dynamic-5c74594ccb-zl9pf   2/2   Running   0      3m
   ```

2. Run the following command to check whether a new disk is mounted to the */data* path:

   ```
   kubectl exec nginx-dynamic-5c74594ccb-zl9pf df | grep data
   ```

   The following output is returned:

   ```
   /dev/vdh      20511312   45080  20449848   1% /data
   ```

3. Run the following command to query the files in the */data* path:

   ```
   kubectl exec nginx-dynamic-5c74594ccb-zl9pf ls /data
   ```

   The following output is returned:

   ```
   lost+found
   ```

4. Run the following command to create the *dynamic* file in the */data* path:

   ```
   kubectl exec nginx-dynamic-5c74594ccb-zl9pf touch /data/dynamic
   ```

5. Run the following command to query the files in the */data* path:

   ```
   kubectl exec nginx-dynamic-5c74594ccb-zl9pf ls /data
   ```

   The following output is returned:

   ```
   dynamic
   lost+found
   ```

6. Run the following command to delete the pod `nginx-dynamic-5c74594ccb-zl9pf`:

   ```
   kubectl delete pod nginx-dynamic-5c74594ccb-zl9pf
   ```

   The following output is returned:

   ```
   pod "nginx-dynamic-5c74594ccb-zl9pf" deleted
   ```

7. Open another kubectl command-line interface (CLI) and run the following command to query how the pod is deleted and recreated:

   ```
   kubectl get pod -w -l app=nginx
   ```

   The following output is returned:

```
NAME                     READY  STATUS  RESTARTS  AGE
nginx-dynamic-5c74594ccb-zl9pf   2/2   Running  0     6m48s
nginx-dynamic-5c74594ccb-zl9pf  2/2  Terminating  0   7m32s
nginx-dynamic-5c74594ccb-45sd4  0/2  Pending  0   0s
nginx-dynamic-5c74594ccb-45sd4  0/2  Pending  0   0s
nginx-dynamic-5c74594ccb-45sd4  0/2  Init:0/1  0   0s
nginx-dynamic-5c74594ccb-zl9pf  0/2  Terminating  0   7m32s
nginx-dynamic-5c74594ccb-zl9pf  0/2  Terminating  0   7m33s
nginx-dynamic-5c74594ccb-zl9pf  0/2  Terminating  0   7m33s
nginx-dynamic-5c74594ccb-45sd4  0/2  PodInitializing  0   5s
nginx-dynamic-5c74594ccb-45sd4  2/2  Running  0   22s
```

8. Run the following command to query the name of the recreated pod:

   ```
   kubectl get pod
   ```

   The following output is returned:

   ```
   NAME                     READY  STATUS   RESTARTS  AGE
   nginx-dynamic-5c74594ccb-45sd4   2/2   Running   0     2m
   ```

9. Run the following command to verify that the *dynamic* file in the */data* path is not deleted. This indicates that the disk data is persistently stored.

   ```
   kubectl exec nginx-dynamic-5c74594ccb-45sd4 ls /data
   ```

   The following output is returned:

   ```
   dynamic
   lost+found
   ```

9.4.7. Use FlexVolume to dynamically expand a disk volume in ACK

In Kubernetes 1.16, the feature to dynamically expand a disk volume is in public preview. Container Service for Kubernetes (ACK) allows you to dynamically expand a disk volume by using FlexVolume in Kubernetes 1.16 and later. This topic describes how to dynamically expand a disk volume by using FlexVolume.

Context

The expansion includes the expansion of the **disk** size and the expansion of the **file system**. Both expansions can be performed without the need to stop the application where the disk volume is mounted (the disk and file system are still mounted during the expansion). However, to ensure the stability of the file system and application, we recommend that you stop the application, unmount the persistent volume (PV) from the directory, and then expand the PV.

Terms

- Automatic expansion
  You only need to modify the size specified in the persistent volume claim (PVC). The modification is automatically implemented to expand the corresponding PV and disk.
- Manual expansion
  You need to manually expand the PV and run the `resize2fs` command to expand the file system.
- Expansion without stopping the application
  You need to expand the mounted disk and file system without stopping the application.
- Expansion after stopping the application

You need to expand the mounted disk volume and file system after stopping the application.

Kubernetes 1.16 and later allow you to expand a PV without stopping the pod to which the PV is mounted.

Instruction

- **Limits**
  You can dynamically expand only disks that are smaller than 2,000 GiB.
- **Data backup**
  To avoid data loss caused by errors during the expansion, we recommend that you first create a snapshot for the PV to back up the disk data.
- **Scenarios**
  - Dynamic expansion is applicable to only dynamically provisioned PVs, which are mounted by using PVCs that contain the StorageClassName parameter.
  - ACK does not allow you to expand inline disk volumes. Inline disk volumes are not created by using PVs and PVCs.
  - ACK does not allow you to dynamically expand volumes that are associated with basic disks.
  - Specify **AllowVolumeExpansion: True** for the StorageClass. The AllowVolumeExpansion parameter is automatically set to True for StorageClasses that are created by ACK. For StorageClasses that are manually created, you must manually set the AllowVolumeExpansion parameter to True.
- **Plug-in version**
  Make sure that the FlexVolume or CSI plug-in is upgraded to the latest version.

Grant the ResizeDisk permission to the RAM role of the cluster

Before you dynamically expand a mounted disk without stopping the application, you must grant the ResizeDisk permission to the **RAM role** of the cluster. Perform the following steps based on the cluster type and the volume plug-in that is used:

Dedicated Kubernetes cluster that uses CSI

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane, click **Cluster Information**.
5. Click the **Cluster Resources** tab and click the hyperlink next to **Master RAM Role**.
6. In the RAM console, grant the ResizeDisk permission to the RAM role. For more information, see Modify a custom policy.

```
"Action": [
  "ecs:AttachDisk",
  "ecs:DetachDisk",
  "ecs:DescribeDisks",
  "ecs:CreateDisk",
  "ecs:CreateSnapshot",
  "ecs:DeleteDisk",
  "ecs:ResizeDisk",
  "ecs:CreateNetworkInterface",
  "ecs:DescribeNetworkInterfaces",
  "ecs:AttachNetworkInterface",
  "ecs:DetachNetworkInterface",
  "ecs:DeleteNetworkInterface",
  "ecs:DescribeInstanceAttribute",
  "ecs:AssignPrivateIpAddresses",
  "ecs:UnassignPrivateIpAddresses",
  "ecs:DescribeInstances"
```

Managed or dedicated Kubernetes cluster that uses FlexVolume

Perform the preceding Step 1 to Step 4 and click the hyperlink next to **Master RAM Role**.

Expand a disk volume without the need to restart the pod

1. Connect to Kubernetes clusters by using kubectl.

In this example, the pod that you want to manage is in the following state
Run the following command to query the information about the pod:

```
kubectl get pod
```

The following output is returned:

```
web-0     1/1    Running  0      42s
```

Run the following command to view the mounting details of the pod:

```
kubectl exec web-0 df /data
```

The following output is returned:

```
Filesystem    1K-blocks  Used Available Use% Mounted on
/dev/vdb      20511312 45080  20449848   1% /data
```

Run the following command to query the information about the persistent volume claim (PVC):

```
kubectl get pvc
```

The following output is returned:

```
NAME        STATUS  VOLUME           CAPACITY  ACCESS MODES  STORAGECLASS         AGE
disk-ssd-web-0  Bound   d-wz9hpoifm43yn9zie6gl  20Gi     RWO       alicloud-disk-available  57s
```

Run the following command to query the information about the persistent volume (PV):

```
kubectl get pv
```

The following output is returned:

```
NAME            CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM           STORAGECLASS
REASON  AGE
d-wz9hpoifm43yn9zie6gl  20Gi     RWO       Delete       Bound    default/disk-ssd-web-0  alicloud-disk-av
ailable       65s
```

2. Make sure that the requirements that are described in the Instruction section are met. After the requirements are met, run the following command to expand the volume:

```
kubectl patch pvc disk-ssd-web-0 -p '{"spec":{"resources":{"requests":{"storage":"30Gi"}}}}'
```

Wait one minute and then perform the following steps to check whether the volume is expanded.
Run the following command to query the information about the PV:

```
kubectl get pv d-wz9hpoifm43yn9zie6gl
```

The following output is returned:

```
NAME            CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM           STORAGECLASS       R
EASON  AGE
d-wz9hpoifm43yn9zie6gl  30Gi     RWO       Delete       Bound   default/disk-ssd-web-0  alicloud-disk-availa
ble       5m23s
```

Run the following command to query the information about the PVC:

```
kubectl get pvc
```

The following output is returned:

```
NAME        STATUS  VOLUME           CAPACITY  ACCESS MODES  STORAGECLASS        AGE
disk-ssd-web-0  Bound   d-wz9hpoifm43yn9zie6gl  30Gi     RWO       alicloud-disk-available  5m10s
```

Run the following command to view the mounting details of the pod:

```
kubectl exec web-0 df /data
```

The following output is returned:

```
Filesystem    1K-blocks  Used Available Use% Mounted on
/dev/vdb      30832548 45036  30771128   1% /data
```

To expand a disk volume without the need to restart the pod, you need only to run the preceding command.

Restart the pod to which a disk volume is mounted and then expand the disk volume

1. Connect to a Kubernetes cluster by using kubectl. For more information, see Connect to Kubernetes clusters by using kubectl.

   In this example, the pod that you want to manage is in the following state.
   Run the following command to query information about the pod:

   ```
   kubectl get pod
   ```

   Expected output:

   ```
   web-0     1/1   Running  0      42s
   ```

   Run the following command to view the mounting state of the pod:

   ```
   kubectl exec web-0 df /data
   ```

   Expected output:

   ```
   Filesystem    1K-blocks  Used Available Use% Mounted on
   /dev/vdb      20511312 45080  20449848   1% /data
   ```

   Run the following command to query information about the PVC:

   ```
   kubectl get pvc
   ```

   Expected output:

   ```
   NAME        STATUS  VOLUME           CAPACITY  ACCESS MODES  STORAGECLASS        AGE
   disk-ssd-web-0  Bound   d-wz9hpoifm43yn9zie6gl  20Gi     RWO       alicloud-disk-available  57s
   ```

   Run the following command to query information about the PV:

   ```
   kubectl get pv
   ```

   Expected output:

   ```
   NAME            CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS   CLAIM           STORAGECLASS
   REASON  AGE
   d-wz9hpoifm43yn9zie6gl  20Gi     RWO       Delete      Bound    default/disk-ssd-web-0  alicloud-disk-av
   ailable       65s
   ```

2. Run the following command to view the scheduling information about the PV:

```
kubectl get pv d-wz9g2j5qbo37r2lamkg4 -oyaml | grep failure-domain.beta.kubernetes.io/zone
  failure-domain.beta.kubernetes.io/zone: cn-shenzhen-e
```

3. Replace the value of the zone field in the label of the scheduled resource. Then, the pod to which the PV is mounted cannot be scheduled.

   For example, you can change the value of the zone field in the label from cn-shenzhen-e to cn-shenzhen-e-nozone.

   ```
   kubectl label pv d-wz9g2j5qbo37r2lamkg4 failure-domain.beta.kubernetes.io/zone=cn-shenzhen-e-nozone --overwrite
   persistentvolume/d-wz9g2j5qbo37r2lamkg4 labeled
   ```

4. Restart the pod.

   The pod scheduling setting is modified. Therefore, the pod temporarily remains in the Pending state. Run the following command to delete the pod:

   ```
   kubectl delete pod web-0
   ```

   Run the following command to query the information about the pod:

   ```
   kubectl get pod
   ```

   The following output is returned:

   ```
   web-0  0/1   Pending  0     27s
   ```

5. Run the following command to expand the volume:

   ```
   kubectl patch pvc disk-ssd-web-0 -p '{"spec":{"resources":{"requests":{"storage":"30Gi"}}}}'
   ```

6. Restart the pod by changing the zone field in the label to the previous setting. In this example, change the value from cn-shenzhen-e-nozone to cn-shenzhen-e.

   ```
   kubectl label pv d-wz9g2j5qbo37r2lamkg4 failure-domain.beta.kubernetes.io/zone=cn-shenzhen-e --overwrite
   persistentvolume/d-wz9g2j5qbo37r2lamkg4 labeled
   ```

   Wait one minute and then perform the following steps to check whether the volume is expanded. Run the following command to query the information about the pod:

   ```
   kubectl get pod
   ```

   The following output is returned:

   ```
   web-0  1/1   Running  0     3m23s
   ```

   Run the following command to query the information about the PVC:

   ```
   kubectl get pvc
   ```

   The following output is returned:

   ```
   disk-ssd-web-0  Bound   d-wz9g2j5qbo37r2lamkg4  30Gi     RWO       alicloud-disk-available  17m
   ```

   Run the following command to query the information about the PV:

   ```
   kubectl get pv d-wz9g2j5qbo37r2lamkg4
   ```

   The following output is returned:

```
d-wz9g2j5qbo37r2lamkg4  30Gi    RWO      Delete     Bound   default/disk-ssd-web-0  alicloud-disk-availa
ble       17m
```

Run the following command to view the mounting details of the pod:

```
kubectl exec web-0 df /data
```

The following output is returned:

```
/dev/vdb      30832548 45036  30771128  1% /data
```

The result indicates that the disk volume is expanded from 20 GiB to 30 GiB.

9.4.8. Manually expand a disk volume

In a Container Service for Kubernetes (ACK) cluster that runs Kubernetes earlier than 1.16, you cannot set the cluster to automatically expand disk volumes. You must manually expand disk volumes. This topic describes how to manually expand a disk volume.

Context

Terms

- Automatic expansion
  You need only to modify the volume size specified in a persistent volume claim (PVC). The modification is automatically applied to expand the corresponding disk volume and the file system.
- Manual expansion
  You must manually expand the disk volume and run the `resize2fs` command to expand the file system.
- Expansion without service interruption
  You can expand the disk volume and file system without the need to stop the application.
- Expansion with service interruption
  You must first stop the application, expand the disk volume and file system, and then start the application again.

To expand a disk volume, perform the following operations:

- Expand the size of the disk: You must perform this operation in the Elastic Compute Service (ECS) console.
- Expand the file system: You must connect to the ECS instance to which the disk is mounted and then perform this operation.
- Modify the volume size specified in the persistent volume (PV) and PVC. This operation is not supported in current versions.

> **Note**
> Disk volumes cannot be automatically expanded in earlier Kubernetes versions due to the following reasons. We recommend that you upgrade Kubernetes to the latest version and then modify the volume size specified in the PV and PVC to automatically expand the disk volume.
>
> - The procedure for modifying the volume size specified in the PV and PVC varies based on the Kubernetes version.
> - The volume size specified in the PV and PVC does not affect the use of the underlying storage device. This means that if the volume size specified in a PV and a PVC is 20 GiB and the sizes of the disk and file system are 30 GiB, the application can use at most 30 GiB of storage.

To ensure stability, Container Service for Kubernetes (ACK) allows you to use the following methods to expand a disk volume:

- Manually expand a disk volume without service interruption: If the I/O throughput of the disk is high when

you expand the file system, an I/O error may occur in the file system. You do not need to restart the application if you choose this method.

- Manually expand a disk volume with service interruption: After the application is stopped, the disk I/O operations are stopped. This ensures data security when you expand the file system. Your service will be temporarily interrupted if you choose this method.

Usage notes

- **Limits**
  You can expand only disk volumes that are smaller than 2,000 GiB.
- **Data backup**
  Before you expand a disk volume, you must back up the disk data by creating a snapshot of the disk. This prevents data loss when you expand the disk volume.
- **Scenarios**
  You cannot automatically expand disk volumes in the following scenarios:
  - The Kubernetes version of the cluster is earlier than 1.16.
  - The PV is a statically provisioned disk volume.

Examples

A stateful application named web is used in this example to demonstrate how to expand a disk volume by using the preceding two methods. Perform the following operations to query information about the disk:

- Run the following command to query the pods that are provisioned for the web application:

  ```
  kubectl get pod | grep web
  ```

  Expected output:

  ```
  NAME            READY  STATUS  RESTARTS  AGE
  web-0           1/1    Running 0     11h
  web-1           1/1    Running 0     11h
  ```

- Run the following command to query the PVCs that are created for the web application:

  ```
  kubectl get pvc | grep web
  ```

  Expected output:

  ```
  NAME       STATUS  VOLUME    CAPACITY  ACCESS MODES  STORAGECLASS     AGE
  disk-ssd-web-0  Bound  d-0jlhaq***  20Gi    RWO       alicloud-disk-essd  11h
  disk-ssd-web-1  Bound  d-0jl0j5***  20Gi    RWO       alicloud-disk-essd  11h
  ```

- Run the following command to query the PVs that are created for the web application:

  ```
  kubectl get pv | grep web
  ```

  Expected output:

  ```
  NAME     CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM          STORAGECLASS     REASON
  AGE
  d-0jl0j5***  20Gi    RWO      Delete      Bound   default/disk-ssd-web-1  alicloud-disk-essd      11h
  d-0jlhaq***  20Gi    RWO      Delete      Bound   default/disk-ssd-web-0  alicloud-disk-essd      11h
  ```

The output indicates that two disks named `d-0jl0j5***` and `d-0jlhaq***` are used by the web application. Both disks are 20 GiB in size. The disks are mounted to two pods separately.

For more information about how to deploy a stateful application, see Use a StatefulSet to create a stateful application.

Method 1: Expand the disk volume without service interruption

Find the corresponding disk based on the PV information, manually expand the disk, and then connect to the node to which the disk is mounted and expand the file system. The following example demonstrates how to expand both disks to 30 GiB.

Step 1: Expand the disks

1. Log on to the ECS console.
2. In the left-side navigation pane, choose **Storage & Snapshots > Disks**.
3. Find the disks named *d-0jl0j5**** and *d-0jlhaq**** and check the states of the disks. Then, choose **More > Resize Disk** in the **Actions** column for each disk.
4. On the **Resize Disks** page, select **Online Resizing**, and enter the size to which you want to expand the disk in the Size after Resize section. In this example, set the size to 30 GiB.

   > **Note** The specified value cannot be smaller than the current disk size.

5. Confirm the disk expansion fee. Read and select ECS Service Terms, and then click **Confirm**.

   For more information, see Resize disks online for Linux instances.

Step 2: Expand the file systems

After the disks are expanded, you must expand the file systems. Otherwise, the storage that the application can use is still 20 GiB.

> **Notice** This step is intended for unpartitioned disks that are used in Kubernetes. We recommend that you do not use partitioned disks in Kubernetes.
>
> - If an unpartitioned disk is mounted as a PV, you cannot manually create partitions for the disk. Otherwise, the file system may be damaged and data loss may occur.
> - If a partitioned disk is mounted as a PV, you must expand the file system after you can expand the partitioned disk. For more information, see Step 3: View the disk partitions and Step 4: Resize partitions.

1. View the ECS instances to which the disks are mounted.
   i. Log on to the ECS console.
   ii. In the left-side navigation pane, choose **Storage & Snapshots > Disks**.
   iii. Find the disks named *d-0jl0j5**** and *d-0jlhaq****, and click the name of each disk.
   iv. On the **Details** page, click **Attached To** in the **Attaching Information** section.
   v. On the **Instance Details** tab, view **Network Information** about the ECS instance.

   > **Note** You can also view the ECS instances to which the disks are mounted in the ACK console. For more information, see View pods.

2. Connect to the ECS instance to which the disk is mounted and obtain the driver letter of the disk.

   For more information about how to connect to an ECS instance, see Methods used to connect to ECS instances.
   You can use the following methods to obtain the driver letter of the disk.
   - Obtain the driver letter of the disk.
   - Run the following command to query the driver letter of the disk named *d-0jlhaq****:

```
# Query {pv-name}
mount |grep d-0jlhaq***
```

Expected output:

```
/dev/vdc on /var/lib/kubelet/plugins/kubernetes.io/csi/pv/d-0jlhaq***/globalmount type ext4 (rw,relatime
)
/dev/vdc on /var/lib/kubelet/pods/a26d174f-***/volumes/kubernetes.io~csi/d-0jlhaq***/mount type ext4 (
rw,relatime)
```

The output indicates that the driver letter of the `d-0jlhaq***` disk is */dev/vdc*.

3. Run the following command to expand the file system:

```
resize2fs /dev/vdc
```

> **Note** */dev/vdc* is the driver letter obtained in Step 2.

Expected output:

```
resize2fs 1.43.5 (04-Aug-2017)
Filesystem at /dev/vdc is mounted on /var/lib/kubelet/plugins/kubernetes.io/csi/pv/d-0jlhaq***/globalmount
; on-line resizing required
old_desc_blocks = 3, new_desc_blocks = 4
The filesystem on /dev/vdc is now 7864320 (4k) blocks long.
```

4. Run the following command to check whether the file system is expanded:

```
lsblk /dev/vdc
```

Expected output:

```
NAME MAJ:MIN  RM SIZE RO TYPE MOUNTPOINT
vdc  254:32   0  30G  0 disk /var/lib/kubelet/pods/a26d174f-***/volumes/kubernetes.io~csi/d-0jlhaq***/mount
```

The output indicates that the size of the *vdc* file system is expanded to 30 GiB.

Method 2: Expand a disk volume with service interruption

You can stop the application by deleting the StatefulSet or set the value of Replica to 0. Then, you can manually expand the disks and restart the application. The following example demonstrates how to expand both disks to 30 GiB.

Step 1: Delete the pods that are provisioned for the application

1. Run the following command to scale the number of pods to 0:

```
kubectl scale sts web --replicas=0
```

Expected output:

```
statefulset.apps/web scaled
```

2. Run the following command to check whether the pods are deleted:

```
kubectl get pod | grep web
```

No output is returned. This indicates that the web application is stopped.

Step 2: Expand the disks

1. Log on to the ECS console.

2. In the left-side navigation pane, choose **Storage & Snapshots > Disks**.
3. Find the disks named *d-0jl0j5**** and *d-0jlhaq**** and check the states of the disks. Then, choose **More > Resize Disk** in the **Actions** column for each disk.
4. On the **Resize Disks** page, select a method to resize the disk, and specify the size to which you want to expand the disk.
   - If the disk is in the **Unattached** state, do not select **Online Resizing** on the **Resize Disks** page. Specify the size to which you want to expand the disk in the Size after Resize section. In this example, set the value to 30 GiB.
   - If the disk is in the **In Use** state, select **Online Resizing** on the **Resize Disks** page, and specify the size to which you want to expand the disk in the Size after Resize section.

   > **Note** The specified value cannot be smaller than the current disk size.

5. Confirm the disk expansion fee. Read and select ECS Service Terms, and then click **Confirm**.

   For more information, see Resize disks online for Linux instances.

Step 3: Expand the file systems

After the disks are expanded, you must expand the file systems. Otherwise, the storage that the application can use is still 20 GiB.

> **Notice** This step is intended for unpartitioned disks that are used in Kubernetes. We recommend that you do not use partitioned disks in Kubernetes.
>
> - If an unpartitioned disk is mounted as a PV, you cannot manually create partitions for the disk. Otherwise, the file system may be damaged and data loss may occur.
> - If a partitioned disk is mounted as a PV, you must expand the file system after you can expand the partitioned disk. For more information, see Step 3: View the disk partitions and Step 4: Resize partitions.

1. (Optional)Mount the disks to an ECS instance.

   > **Note** You must mount the disks to an ECS instance before you can expand the file systems.

   i. Log on to the ECS console.
   ii. In the left-side navigation pane, choose **Storage & Snapshots > Disks**.
   iii. Find the disk that is in the **Unattached** state and choose **More > Attach** in the **Actions** column.

iv. In the **Attach Disk** dialog box, select an ECS instance from the drop-down list, and configure the settings that correspond to releasing disks.

Parameter	Description
Target Instance	Select the ECS instance to which you want to mount the disk.
Release Disk with Instance	If you select this option, the disk is automatically released when the ECS instance to which it is mounted is released. If you do not select this option, the disk is retained when the ECS instance to which it is mounted is released. **Note** If the disk that you want to mount is a system disk that you unmounted from another ECS instance, the ECS instance specified by **Release Disk with Instance** refers to the ECS instance from which you unmounted the disk.
Delete Automatic Snapshots While Releasing Disk	If you select this option, snapshots that are automatically created for the disk are released together with disk. To retain the snapshots, do not select this option.

v. Click **Attach**.

If the status of the disk becomes **In Use**, the disk is attached.

2. Connect to the ECS instance to which the disk is mounted and obtain the driver letter of the disk.

   For more information about how to connect to an ECS instance, see Methods used to connect to ECS instances.

   - Run the following command to obtain the driver letter of the disk:

     ```
     for device in `ls /sys/block | grep vd`; do
      cat /sys/block/$device/serial | grep 0jlhaq*** && echo $device;
     done
     ```

     Note The ID of the expanded disk is `d-0jlhaq***`. `0jlhaq***` is the string that follows `d-`.

   - (Optional)If you cannot obtain the driver letter of the disk by running the preceding command, perform the following operations:

     a. Unmount the disk and run the `ls /dev/vd*` command to query the list of disks.

     b. Mount the disk and run the `ls /dev/vd*` command to query the list of disks.

     c. Compare the lists that are returned. The disk that appears only in the second list is the one that you mounted.

3. Run the following command to expand the file system:

   ```
   resize2fs /dev/vdb
   ```

   Note */dev/vdc* is the driver letter of the disk obtained in the Step 2.

   Expected output:

```
resize2fs 1.43.5 (04-Aug-2017)
Resizing the filesystem on /dev/vdb to 7864320 (4k) blocks.
The filesystem on /dev/vdb is now 7864320 (4k) blocks long.
```

4. Check whether the file system is expanded.
   i. Run the following command to create a temporary folder named */mnt/disk/* and mount the disk to the folder:

   ```
   mkdir /mnt/disk
   mount /dev/vdb /mnt/disk/
   ```

   ii. Run the following command to query the size of the specified file system:

   ```
   df /mnt/disk/
   ```

   Expected output:

   ```
   Filesystem    1K-blocks  Used Available Use% Mounted on
   /dev/vdb      30832548 45036  30771128   1% /mnt/disk
   ```

   The output indicates that the */dev/vdb* folder can use at most 30 GiB of storage. This indicates that the file system is expanded.

   iii. Run the following command to unmount the disk from the temporary folder:

   ```
   umount /mnt/disk
   ```

Step 4: Restart the application

1. Run the following command to scale the number of pods to 2:

   ```
   kubectl scale sts web --replicas=2
   ```

   Expected output:

   ```
   statefulset.apps/web scaled
   ```

2. Run the following command to check whether the pods are deleted:

   ```
   kubectl get pod | grep web
   ```

   Expected output:

   ```
   NAME    READY  STATUS   RESTARTS  AGE
   web-0   1/1    Running  0     74s
   web-1   1/1    Running  0     42s
   ```

3. Run the following command to query the size of the specified file system:

   ```
   kubectl exec web-0 df /data
   ```

   Expected output:

   ```
   Filesystem    1K-blocks  Used Available Use% Mounted on
   /dev/vdb      30832548 45036  30771128   1% /data
   ```

   The output indicates that the size of the */dev/vdb* file system is expanded to 30 GiB.

FAQ

Issue:

What can I do if the following message appears after I run the `resize2fs` command?

```
resize of device /dev/xxx failed: exit status 1 resize2fs output: resize2fs xxx(version)
Please run `e2fsck -f /dev/xxx` first
```

Cause:

The file systems are not consistent, which causes I/O errors.

Solution:

Run the `e2fsck -f /dev/xxx` command and then expand the file systems.

Related information

- Automatically expand a disk volume

9.5. NAS volumes

9.5.1. Use NAS volumes

This topic describes how to mount Network Attached Storage (NAS) file systems to clusters of Container Service for Kubernetes (ACK) as volumes and how to use NAS volumes.

Prerequisites

A NAS file system is created and a mount target is added to the file system. To create a NAS file system and add a mount target, log on to the NAS console. The mount target of the NAS file system and your cluster are deployed in the same virtual private cloud (VPC).

The mount target is in the following format: `055f84ad83-ixxxx.cn-hangzhou.nas.aliyuncs.com` .

Background information

You can mount file systems of Apsara File Storage NAS to ACK clusters in the following ways:

- Mount NAS file systems as static volumes
  - Directly mount NAS file systems as volumes.
  - Use a pair of persistent volume (PV) and persistent volume claim (PVC) to mount NAS file systems.
- Mount NAS file systems as dynamic volumes

Scenarios

- Static volumes
  NAS provides shared storage services. You can mount NAS file systems as static volumes to meet the requirements of diverse scenarios.
- Dynamic volumes
  You can mount NAS file systems as dynamic volumes when you need to use multiple NAS sub-directories for different applications.
  You can also mount NAS file systems as dynamic volumes when you use the StatefulSet controller to deploy applications and want each pod to use a separate NAS volume.

How to mount NAS file systems

We recommend that you read the following information before you mount NAS file systems to ACK clusters:

- Recommended volume plug-in
  We recommend that you use the Flexvolume driver to mount NAS file systems.

The Flexvolume driver is installed by default when you create an ACK cluster in the console. You must make sure that the Flexvolume driver is upgraded to the latest version. For more information, see Upgrade the Flexvolume driver.

- Recommended mounting method
  We recommend that you mount a NAS file system by using a pair of PV and PVC. This makes the NAS file system easier to manage and maintain.
  - For more information about static volumes, see Statically provisioned NAS volumes.
  - For more information about dynamic volumes, see Dynamic NAS volumes.
- Not recommended mounting method
  We recommend that you do not directly mount NAS file systems as volumes. You can use only the Flexvolume driver to mount volumes to ACK clusters. The Network File System (NFS) driver provided by Kubernetes is not supported.

Considerations

- NAS is a shared storage system that provides storage services for multiple pods at a time. A PVC can be shared among multiple pods.
- Do not delete a mount target if the related NAS file system is still mounted. Otherwise, the operating system may become unresponsive.
- After a mount target is created, wait until the mount target is ready for use.
- We recommend that you use NFS v3.
- We recommend that you upgrade Flexvolume to the latest version before you use NAS volumes.
- NAS file systems of Extreme type support only NFS v3. You must specify the nolock parameter when you mount these file systems.

9.5.2. Statically provisioned NAS volumes

You can use the FlexVolume plug-in provided by Alibaba Cloud to use Apsara File Storage NAS in Container Service for Kubernetes (ACK). This topic describes how to use a statically provisioned NAS volume.

Prerequisites

- Upgrade the FlexVolume plug-in to the latest version.
- Connect to Kubernetes clusters by using kubectl.

Context

The FlexVolume plug-in allows you to use a NAS file system as a volume. You can also use a NAS file system to create a PV and a PVC.
The following parameters are included:

- server: the mount target of the NAS volume.
- path: the mounted directory in the NAS file system. You can specify a sub-directory as a volume. If no sub-directory exists, the system automatically creates and mounts a sub-directory. To specify a directory in a NAS Extreme file system, the directory must start with */share*.
- vers: the version of the NFS protocol. Versions 3 and 4.0 are supported. By default, version 3 is used. We recommend that you use version 3. NAS Extreme file systems support only NFS version 3.
- mode: the access permissions on the mounted directory. If the root directory of the NAS file system is specified as the mounted directory, you cannot modify the access permissions. If you set the mode parameter for a directory that stores a large amount of data, the process of mounting the directory to a cluster may require an excessive amount of time or even fail.
- options: the mount parameter. If you do not set this parameter, the default values are `nolock,tcp,noresvport` in version 3 and `noresvport` in version 4.0.

Use a NAS file system as a volume

You can set the volumes field to specify a NAS file system as a volume of pods. Use a *nas-deploy.yaml* file to create a pod.

1. Create the *nas-deploy.yaml* file and copy the following content to the file.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nas-static
 labels:
  app: nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
     - name: nas1
      mountPath: "/data"
   volumes:
   - name: "nas1"
    flexVolume:
     driver: "alicloud/nas"
     options:
      server: "0cd8b4a576-grs79.cn-hangzhou.nas.aliyuncs.com"
      path: "/k8s"
      vers: "3"
      options: "nolock,tcp,noresvport"
```

2. Run the following command to create a pod:

```
kubectl apply -f nas-deploy.yaml
```

Use a NAS file system to create a PV and a PVC

You can use a NAS file system to create a PV and a PVC and associate them with a pod.

1. Create a PV.

   You can create a PV in the ACK console or by using a YAML file.

   - Create a PV by using a YAML file.
     Use the following *nas-pv.yaml* file to create a PV.

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: pv-nas
spec:
 capacity:
  storage: 5Gi
 storageClassName: nas
 accessModes:
  - ReadWriteMany
 flexVolume:
  driver: "alicloud/nas"
  options:
   server: "0cd8b4a576-uih75.cn-hangzhou.nas.aliyuncs.com"
   path: "/k8s"
   vers: "3"
   options: "nolock,tcp,noresvport"
```

- Create a PV in the ACK console
  a. Log on to the ACK console.
  b. In the left-side navigation pane, click **Clusters**.
  c. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column of the cluster.
  d. The Cluster Information page appears. In the left-side navigation pane, click **Persistent Volumes**. The **Persistent Volume Claims** tab appears.
  e. Click the **Persistent Volumes** tab and click **Create**.
  f. In the Create PV dialog box, set the parameters.

Parameter	Description
PV Type	In this example, NAS is selected.
Name	The name of the PV. The name must be unique in the cluster. In this example, pv-nas is used.
Volume Plug-in	In this example, FlexVolume is selected. For more information about volume plug-ins, see Differences between FlexVolume and CSI.
Capacity	The capacity of the PV. The capacity of the PV cannot exceed that of the NAS file system.
Access Mode	By default, ReadWriteMany is selected.
Mount Target Domain Name	The domain name of the mount target that is used to mount the NAS file system to the cluster.

Parameter	Description
Subdirectory	Enter a sub-directory in the NAS file system. The sub-directory must start with a forward slash (/). If this parameter is set, the PV will be mounted to the sub-directory. ■ If the specified sub-directory does not exist, the system automatically creates this sub-directory. ■ If you do not set this parameter, the root directory of the NAS file system is mounted. ■ To specify a sub-directory in a NAS Extreme file system, the sub-directory must start with */share*.
Permissions	The access permissions on the mounted directory. For example, you can set this parameter to 755, 644, or 777. **Note** ■ You can set access permissions on only sub-directories. ■ We recommend that you do not set this parameter when the mounted directory contains a large number of files. Otherwise, it may take a long time to run the chmod command. If the mounted directory is a sub-directory of the NAS file system, the Permissions parameter is optional. ■ By default, the original permissions are used. ■ Take note of the following requirements when you set the permissions: ■ For FlexVolume versions earlier than v1.14.6.15-8d3b7e7-aliyun, use the recursive mode when you change permission settings. The permissions on all files and directories under the mounted directory will be changed. ■ For FlexVolume v1.14.6.15-8d3b7e7-aliyun and later, set the **chmod (Change Mode)** parameter to define how permission changes are applied.

Parameter	Description
chmod (Change Mode)	The change mode of access permissions. Valid values: Non-recursive and Recursive. ■ Non-recursive: The permission changes apply to only the mount directory. The subdirectories and files in the mount directory are not affected. ■ Recursive mode: The permission changes apply to the mounted directory, the subdirectories, and the included files. **Note** If you select the recursive mode for a mounted directory that contains a large number of files, the process of running the chmod command may require an excessive amount of time. The mount or unmount operation may fail. Exercise caution when you perform this operation.
Version	The version of the NFS protocol. We recommend that you use version 3. NAS Extreme file systems support only NFS version 3.
Labels	Add labels to the PV.

g. Click **Create**.

2. Create a PVC.

   Use the following *nas-pvc.yaml* file to create a PVC.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: pvc-nas
spec:
 accessModes:
  - ReadWriteMany
 storageClassName: nas
 resources:
  requests:
   storage: 5Gi
```

3. Create a pod.

   Use the following *nas-pod.yaml* file to create a pod.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nas-static
 labels:
  app: nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
     - name: pvc-nas
      mountPath: /data
   volumes:
    - name: pvc-nas
     persistentVolumeClaim:
      claimName: pvc-nas
```

Related information

- Use NAS volumes

9.5.3. Dynamic NAS volumes

This topic describes how to create a dynamic NAS volume by creating a subdirectory in a NAS file system and mapping the subdirectory to a dynamic persistent volume (PV) for applications.

Prerequisites

The FlexVolume driver is installed in a Container Service for Kubernetes (ACK) cluster. By default, the FlexVolume driver is installed for ACK clusters.
alicloud-nas-controller is deployed. For more information, see Install and upgrade FlexVolume.

Create a dynamic NAS volume

1. Configure a StorageClass.

   The following code block is an example of the StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: alicloud-nas
mountOptions:
- nolock,tcp,noresvport
- vers=3
parameters:
 server: "23a9649583-iaq37.cn-shenzhen.nas.aliyuncs.com:/nasroot1/"
 driver: flexvolume
provisioner: alicloud/nas
reclaimPolicy: Delete
```

Note

- mountOptions: the mount options of the PV. The NAS volume is mounted based on the specified mount options.
- server: the list of mount targets that are used by the PV. The format is *nfsurl1:/path1,nfsurl2:/path2*. When multiple servers are configured, the PV provisioned by this StorageClass uses the servers in a round robin manner. For NAS file systems of Extreme type, the path must start with */share*.
- driver: supports the FlexVolume and NFS drivers. The default driver is NFS.
- reclaimPolicy: the reclaim policy of the PV. We recommend that you set this parameter to Retain.
  - If you set this parameter to Delete, the name of the subdirectory mapped to the PV is automatically changed after you delete the PV. For example, *path-name* is changed to *archived-path-name*.
  - If you want to delete the subdirectory in the NAS file system, set `archiveOnDelete` to *false* in the StorageClass.

2. Use the dynamic NAS volume.

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 labels:
  app: nginx
spec:
 ports:
 - port: 80
  name: web
 clusterIP: None
 selector:
  app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: web
spec:
 serviceName: "nginx"
 replicas: 5
 volumeClaimTemplates:
 - metadata:
    name: html
  spec:
   accessModes:
    - ReadWriteOnce
   storageClassName: alicloud-nas
   resources:
    requests:
     storage: 2Gi
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:alpine
    volumeMounts:
    - mountPath: "/data"
     name: html
```

9.5.4. Use NAS volumes for shared persistent storage

You can use an Apsara File Storage NAS (NAS) volume to persist data and share the data among multiple pods. This topic describes how to use a NAS file system to persist and share data.

Prerequisites

- 创建Kubernetes托管版集群.
- Connect to Kubernetes clusters by using kubectl.
- A NAS file system is created in the NAS File System console. For more information, see Mount an NFS file

system. The NAS file system and the cluster are deployed in the same zone.

- A mount target is added to the NAS file system. For more information, see . The NAS file system and the cluster are deployed in the same virtual private cloud (VPC).

Context

If a NAS file system is mounted on multiple pods, the data in the file system is shared among the pods. The application must be able to synchronize data across all pods when data modifications are made by multiple pods.

Scenarios:

- Your application requires high disk I/O.
- You need a storage service that offers higher read and write throughput than Object Storage Service (OSS).
- You want to share files across hosts. For example, you want to use a NAS file system as a file server.

Procedure

1. Create a NAS file system and create a mount target.
2. Create a persistent volume (PV) and a persistent volume claim (PVC).

The following section describes how to create a PV and a PVC by using the *FlexVolume* plug-in provided by Alibaba Cloud and then mount a NAS file system.

Create a PV

1. Create a file named *pv-nas.yaml*.

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: pv-nas
 labels:
  alicloud-pvname: pv-nas
spec:
 capacity:
  storage: 5Gi
 accessModes:
  - ReadWriteMany
 flexVolume:
  driver: "alicloud/nas"
  options:
   server: "***-**.cn-hangzhou.nas.aliyuncs.com"  #Replace the value with the mount target.
   path: "/k8s1"
   vers: "4.0"
```

Parameter	Description
alicloud-pvname	The name of the PV.
server	The mount target of the NAS file system. To obtain the mount target, log on to the NAS File System console. In the left-side navigation pane, click **File System List**, find the created file system, and then click **Management** in the **Operations** column. Click the **Mounting Use** tab and copy the mount address in the **Mount Target** column.

Parameter	Description
path	The mounted directory of the NAS file system. You can specify a subdirectory of a NAS file system. If no subdirectories exist, the system automatically creates a subdirectory.
vers	The version number of the Network File System (NFS) protocol. This parameter is optional. Valid values: 3 and 4.0. Default value: 3.
mode	The access permissions on the mounted directory. This parameter is optional. By default, this parameter is left empty. **Note** ◦ You are not allowed to grant permissions to access the root directory of a NAS file system. ◦ If you set the mode parameter for a NAS file system that stores a large amount of data, the process of mounting the NAS file system may be time-consuming or even fail. We recommend that you leave this parameter empty.

2. Run the following command to create a PV:

```
kubectl create -f pv-nas.yaml
```

Expected result

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**. Verify that the newly created PV is displayed.

Create a PVC

When you create a PVC of the NAS type, set the `selector` parameter to configure how to select the PV to which the PVC is bound.

1. Create a file named *pvc-nas.yaml*.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: pvc-nas
spec:
 accessModes:
  - ReadWriteMany
 resources:
  requests:
   storage: 5Gi
 selector:
  matchLabels:
   alicloud-pvname: pv-nas
```

2. Run the following command to create a PVC:

```
kubectl create -f pvc-nas.yaml
```

Expected result

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volume Claims**. Verify that the newly created PVC is displayed.

Create an application

1. Create a file named *nas.yaml*.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nas-static
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
     - name: pvc-nas
      mountPath: "/data"
   volumes:
    - name: pvc-nas
     persistentVolumeClaim:
      claimName: pvc-nas
```

2. Run the following command to deploy an application:

```
kubectl create -f nas.yaml
```

Expected result

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Applications** in the **Actions** column.
4. In the left-side navigation pane of the **cluster** details page, choose **Workloads > Deployments**. Verify that the newly created application is displayed.

Verify data sharing

1. Run the following command to query the pods that run the application.

```
kubectl get pod
```

Expected output:

```
NAME                READY  STATUS  RESTARTS  AGE
nas-static-f96b6b5d7-r****   1/1   Running  0     9m
nas-static-f96b6b5d7-w****   1/1   Running  0     9m
```

2. Run the following commands to query the files in the */data* path:

```
kubectl exec nas-static-f96b6b5d7-r**** ls /data
```

Expected output:

```
kubectl exec nas-static-f96b6b5d7-w**** ls /data
```

> **Note** The output indicates that no file exists in the */data* path.

3. Run the following command to create a file named *nas* in the */data* path of a pod:

```
kubectl exec nas-static-f96b6b5d7-r**** touch /data/nas
```

4. Query files in the pods.

   Run the following command to query files in the */data* path of one pod:

```
kubectl exec nas-static-f96b6b5d7-r**** ls /data
```

   Expected output:

```
nas
```

   Run the following command to query files in the */data* path of the other pod:

```
kubectl exec nas-static-f96b6b5d7-w**** ls /data
```

   Expected output:

```
nas
```

> **Note** The file was created in the */data* path of one of the pods. However, you can find the file in the */data* path of both pods. This indicates that the pods share the NAS volume.

Verify data persistence

1. Run the following commands to delete all pods of the application:

```
kubectl delete pod nas-static-f96b6b5d7-r**** nas-static-f96b6b5d7-wthmb
```

   Expected output:

```
pod "nas-static-f96b6b5d7-r****" deleted
pod "nas-static-f96b6b5d7-w****" deleted
```

2. Run the following command to view how the pods are deleted and recreated:

```
kubectl get pod -w -l app=nginx
```

   Expected output:

```
NAME                READY  STATUS       RESTARTS   AGE
nas-static-f96b6b5d7-r****   1/1   Running       0       27m
nas-static-f96b6b5d7-w****    1/1   Running       0       27m
nas-static-f96b6b5d7-r****   1/1   Terminating     0       28m
nas-static-f96b6b5d7-w****    0/1   Pending       0       0s
nas-static-f96b6b5d7-w****    0/1   Pending       0       0s
nas-static-f96b6b5d7-w****    0/1   ContainerCreating  0       0s
nas-static-f96b6b5d7-w****    1/1   Terminating     0       28m
nas-static-f96b6b5d7-n****    0/1   Pending       0       0s
nas-static-f96b6b5d7-n****    0/1   Pending       0       0s
nas-static-f96b6b5d7-n****    0/1   ContainerCreating  0       0s
nas-static-f96b6b5d7-r****    0/1   Terminating     0       28m
nas-static-f96b6b5d7-w****    0/1   Terminating     0       28m
nas-static-f96b6b5d7-r****    0/1   Terminating     0       28m
nas-static-f96b6b5d7-r****    0/1   Terminating     0       28m
nas-static-f96b6b5d7-w****    1/1   Running       0       10s
nas-static-f96b6b5d7-w****    0/1   Terminating     0       28m
nas-static-f96b6b5d7-w****    0/1   Terminating     0       28m
nas-static-f96b6b5d7-n****    1/1   Running       0       17s
```

3. Run the following command to query the newly created pods:

```
kubectl get pod
```

Expected output:

```
NAME                READY  STATUS  RESTARTS  AGE
nas-static-f96b6b5d7-n****   1/1   Running  0     21s
nas-static-f96b6b5d7-w****    1/1   Running  0     21s
```

4. Query files in the pods.

Run the following command to query files in the */data* path of one pod:

```
kubectl exec nas-static-f96b6b5d7-n**** ls /data
```

Expected output:

```
nas
```

Run the following command to query files in the */data* path of the other pod:

```
kubectl exec nas-static-f96b6b5d7-w**** ls /data
```

Expected output:

```
nas
```

> **Note** The *nas* file still exists. This indicates that data is persisted to the NAS volume.

9.6. OSS volumes

9.6.1. Mount OSS volumes

This topic describes how to mount volumes that use Object Storage Service (OSS) buckets in a Container Service for Kubernetes (ACK) cluster.

ACK supports only statically provisioned OSS volumes. Dynamically provisioned OSS volumes are not supported. You can mount statically provisioned OSS volumes by using the following methods:

- Use an OSS bucket as a statically provisioned volume
- Use an OSS bucket to create a PV and a PVC

Prerequisites

An OSS bucket is created in the OSS console. For more information, see Create buckets.

Instruction

The following section describes how to configure a statically provisioned OSS volume:

- An OSS bucket can be shared by multiple pods.
- bucket: You can mount only buckets to clusters. The subdirectories or files in a bucket cannot be mounted to an ACK cluster.
- url: the endpoint of an OSS bucket. If the bucket and the node to which the bucket is mounted are deployed in the same region, you can specify the internal endpoint of an OSS bucket.
- akId: your AccessKey ID.
- akSecret: your AccessKey secret.
- otherOpts: the custom parameters that are used to mount the OSS bucket. The parameters must be in the following format: `-o *** -o ***`.
- To mount an OSS volume, do not specify subpath.
- We recommend that you create a persistent volume (PV) for each application.
- Only the CentOS and Alibaba Cloud Linux 2 operating systems are supported.

Considerations

- OSS is a Filesystem in Userspace (FUSE) file system that can be mounted by using OSSFS. This method is suitable for read operations. For example, you can use this method to read configuration files, video files, and images.
- OSSFS is not suitable for write operations. If you require write operations, we recommend that you use Apsara File Storage NAS (NAS) file systems.
- Compared with FUSE, the file system in a kernel state offers higher stability and performance. We recommend that you use NAS file systems instead of OSS buckets in production environments.
- You can modify parameter configurations to optimize OSSFS performance in caching and permission management. For more information, see FAQ about OSSFS, ossfs/README-CN.md, and FAQ.

9.6.2. Use statically provisioned OSS volumes

This topic describes how to use statically provisioned Object Storage Service (OSS) volumes in Container Service for Kubernetes (ACK).

Use an OSS bucket as a statically provisioned volume

1. Create a file named *oss-deploy.yaml* and copy the following content to the file.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
 name: nginx-oss-deploy
spec:
 replicas: 1
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx-flexvolume-oss
    image: nginx
    volumeMounts:
     - name: "oss1"
      mountPath: "/data"
    livenessProbe:
     exec:
      command:
      - sh
      - -c
      - cd /data
     initialDelaySeconds: 30
     periodSeconds: 30
   volumes:
    - name: "oss1"
     flexVolume:
      driver: "alicloud/oss"
      options:
       bucket: "docker"
       url: "oss-cn-hangzhou.aliyuncs.com"
       akId: ***
       akSecret: ***
       otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

2. Run the following command to create a pod:

```
kubectl apply -f oss-deploy.yaml
```

Use an OSS bucket to create a PV and a PVC

1. Create a persistent volume (PV).

   You can create a PV in the ACK console or by using a YAML file.

   - Create a PV by using a YAML file
     Use the following *oss-pv.yaml* file to create a PV.

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: pv-oss
spec:
 capacity:
  storage: 5Gi
 accessModes:
  - ReadWriteMany
 storageClassName: oss
 flexVolume:
  driver: "alicloud/oss"
  options:
   bucket: "docker"
   url: "oss-cn-hangzhou.aliyuncs.com"
   akId: ***
   akSecret: ***
   otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

- Create a PV in the ACK console
  a. Log on to the ACK console.
  b. In the left-side navigation pane, click **Clusters**.
  c. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
  d. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.
  e. Click the **Persistent Volumes** tab and click **Create**.

f. In the **Create PV** dialog box, set the required parameters.

Parameter	Description
PV Type	In this example, OSS is selected.
Volume Name	The name of the PV that you want to create. The name must be unique in the cluster. In this example, pv-oss is used.
Volume Plug-in	In this example, Flexvolume is selected. For more information about volume plug-ins, see Differences between FlexVolume and CSI.
Capacity	The capacity of the PV.
Access Mode	Default value: ReadWriteMany.
AccessKey ID	The AccessKey pair that is required to access the OSS bucket.
AccessKey Secret	
Optional Parameters	Enter custom parameters in the format of `-o ** * -o ***`.
Bucket ID	The name of the OSS bucket that you want to mount. Click **Select Bucket**. In the dialog box that appears, select the OSS bucket that you want to mount and click **Select**.
Endpoint	Select **Public Endpoint** if the OSS bucket and the Elastic Compute Service (ECS) instances in the cluster are deployed in different regions. Select **Internal Endpoint** if the OSS bucket is deployed in a classic network.
Label	Add labels to the PV.

g. Click **Create**.

2. Create a PVC.

   Use the following *oss-pvc.yaml* file to create a PVC.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: pvc-oss
spec:
 storageClassName: oss
 accessModes:
  - ReadWriteMany
 resources:
  requests:
   storage: 5Gi
```

3. Create a pod.

   Use the following *oss-deploy.yaml* file to create a pod.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: oss-static
 labels:
  app: nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
     - name: pvc-oss
      mountPath: "/data"
    livenessProbe:
     exec:
      command:
      - sh
      - -c
      - cd /data
     initialDelaySeconds: 30
     periodSeconds: 30
   volumes:
   - name: pvc-oss
    persistentVolumeClaim:
     claimName: pvc-oss
```

Use a Secret to provide AccessKey information

1. Run the following command to create a Secret:

   ```
   kubectl create secret generic osssecret --from-literal=akId='111111' --from-literal=akSecret='2222222' --type=alicloud/oss -n default
   ```

   - osssecret: The name of the Secret.
   - akId: The AccessKey ID.
   - akSecret: The AccessKey secret.
   - type: Set this parameter to alicloud/oss. The Secret and the pod that uses the Secret must belong to the same namespace.

2. Use the Secret in a PV.

   Specify the Secret in the secretRef field of the PV.

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: pv-oss
spec:
 capacity:
  storage: 5Gi
 accessModes:
  - ReadWriteMany
 storageClassName: oss
 flexVolume:
  driver: "alicloud/oss"
  secretRef:
   name: "osssecret"
  options:
   bucket: "docker"
   url: "oss-cn-hangzhou.aliyuncs.com"
   otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

3. Use the Secret in a volume.

   Specify the Secret in the secretRef field of the volume.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
 name: nginx-oss-deploy1
spec:
 replicas: 3
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx-flexvolume-oss
    image: nginx
    volumeMounts:
     - name: "oss1"
      mountPath: "/data"
      subPath: "hello"
   volumes:
    - name: "oss1"
     flexVolume:
      driver: "alicloud/oss"
      secretRef:
       name: "osssecret"
      options:
       bucket: "aliyun-docker"
       url: "oss-cn-hangzhou.aliyuncs.com"
       otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

> **Note** When you use a Secret to configure the AccessKey pair, the Secret and the pod that uses the Secret must belong to the same namespace.

9.6.3. Use OSS volumes for persistent storage

When a node that hosts running containers fails, stateful applications may lose the business data stored in the containers. This issue can be resolved by using persistent storage. This topic describes how to use an Object Storage Service (OSS) volume to persist data.

Background information

OSS is a secure, cost-effective, and highly reliable cloud storage service provided by Alibaba Cloud. You can mount an OSS volume on multiple pods of a Container Service for Kubernetes (ACK) cluster.

Scenarios

- Average requirements on disk I/O
- Sharing of data, including configuration files, images, and small video files

Procedure

1. Create an OSS bucket.
2. Obtain an **AccessKey ID** and **AccessKey secret**.
3. Create a persistent volume (PV) and a persistent volume claim (PVC).

Prerequisites

- 创建Kubernetes托管版集群.
- Connect to Kubernetes clusters by using kubectl.
- An OSS bucket is created in the OSS console. For more information, see Create buckets.

Precautions

- kubelet and the OSSFS driver may be restarted when the ACK cluster is upgraded. As a result, the mounted OSS directory becomes unavailable. In this case, you must recreate the pods on which the OSS volume is mounted. You can add health check settings in the YAML file to restart the pods and remount the OSS volume when the OSS directory becomes unavailable.
- If your ACK cluster is of the latest Kubernetes version, the preceding issue is fixed.

Create a PV

1. Create a file named *pv-oss.yaml*.

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: pv-oss
 labels:
  alicloud-pvname: pv-oss
spec:
 capacity:
  storage: 5Gi
 accessModes:
  - ReadWriteMany
 storageClassName: oss
 flexVolume:
  driver: "alicloud/oss"
  options:
   bucket: "docker"                //Replace the value with the bucket name.
   url: "oss-cn-hangzhou.aliyuncs.com"   //Replace the value with the endpoint of the OSS bucket.
   akId: "***"                  //Replace the value with the AccessKey ID.
   akSecret: "***"                //Replace the value with the AccessKey secret.
   otherOpts: "-o max_stat_cache_size=0 -o allow_other"  //Replace the value with custom parameter values
.
```

Parameter description

- `alicloud-pvname`: the name of the PV. This value must be used in the `selector` field of the PVC that is associated with the PV.
- `bucket`: the name of the OSS bucket. Only OSS buckets can be mounted to the ACK cluster. You cannot mount the subdirectories or files in an OSS bucket to the ACK cluster.
- `url`: the endpoint of the OSS bucket. For more information, see Regions and endpoints. To obtain the endpoint, log on to the OSS console. In the left-side navigation pane, find the OSS bucket that you want to manage. On the **Overview** page, find the **Domain Names** section and view the endpoint of the OSS bucket in the **Endpoint** column.
- `akId`: the AccessKey ID. Log on to the ACK console, move the pointer over the  icon in the upper-right corner of the page and select **AccessKey Management** from the shortcut menu. On the page that appears, create an **AccessKey ID** and an **AccessKey secret**.
- `akSecret`: the AccessKey secret. To obtain the AccessKey secret, perform the steps described in `akId`.
- `otherOpts`: the custom parameters that are used to mount the OSS bucket. The parameters must be in the following format: `-o *** -o ***`.

2. Run the following command to create a PV:

```
kubectl create -f pv-oss.yaml
```

Expected result

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**. Verify that the newly created PV is displayed.

Create a PVC

Create a PVC of the OSS type. Set the `selector` parameter to configure how to select the PV to which the PVC is bound. Set the `storageClassName` parameter to bind the PVC with the PV of the OSS type.

1. Create a file named *pvc-oss.yaml*.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: pvc-oss
spec:
 accessModes:
  - ReadWriteMany
 storageClassName: oss
 resources:
  requests:
   storage: 5Gi
 selector:
  matchLabels:
   alicloud-pvname: pv-oss
```

2. Run the following command to create a PVC:

```
kubectl create -f pvc-oss.yaml
```

Expected result

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volume Claims**. Verify that the newly created PVC is displayed.

Create an application

1. Create a file named *oss-static.yaml*.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: oss-static
 labels:
  app: nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
     - name: pvc-oss
      mountPath: "/data"
     - name: pvc-oss
      mountPath: "/data1"
    livenessProbe:
     exec:
      command:
      - sh
      - -c
      - cd /data
     initialDelaySeconds: 30
     periodSeconds: 30
   volumes:
    - name: pvc-oss
     persistentVolumeClaim:
      claimName: pvc-oss
```

Note For more information about how to set `livenessProbe` to configure health checks, see Mount OSS volumes.

2. Run the following command to deploy an application:

```
kubectl create -f oss-static.yaml d
```

Expected result

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Applications** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**. Verify that the newly created application is displayed.

Verify data persistence

1. Run the following command to query the pods that run the application:

```
kubectl get pod
```

Expected output:

```
NAME                READY  STATUS  RESTARTS  AGE
oss-static-66fbb85b67-dqbl2   1/1   Running  0     1h
```

2. Run the following command to query the files in the */data* path:

```
kubectl exec oss-static-66fbb85b67-dqbl2 ls /data | grep tmpfile
```

> **Note** The output indicates that no file exists in the */data* path.

3. Run the following command to create a file named *tmpfile* in the */data* path:

```
kubectl exec oss-static-66fbb85b67-dqbl2 touch /data/tmpfile
```

4. Run the following command to query the files in the */data* path:

```
kubectl exec oss-static-66fbb85b67-dqbl2 ls /data | grep tmpfile
```

Expected output:

```
tmpfile
```

5. Run the following command to delete the pod named *oss-static-66fbb85b67-dqbl2*:

```
kubectl delete pod oss-static-66fbb85b67-dqbl2
```

Expected output:

```
pod "oss-static-66fbb85b67-dqbl2" deleted
```

6. Open another kubectl command-line interface (CLI) and run the following command to view how the pod is deleted and recreated:

```
kubectl get pod -w -l app=nginx
```

Expected output:

```
NAME            READY  STATUS      RESTARTS  AGE
oss-static-66fbb85b67-dqbl2  1/1   Running      0     78m
oss-static-66fbb85b67-dqbl2  1/1   Terminating    0     78m
oss-static-66fbb85b67-zlvmw  0/1   Pending      0     <invalid>
oss-static-66fbb85b67-zlvmw  0/1   Pending      0     <invalid>
oss-static-66fbb85b67-zlvmw  0/1   ContainerCreating 0     <invalid>
oss-static-66fbb85b67-dqbl2  0/1   Terminating    0     78m
oss-static-66fbb85b67-dqbl2  0/1   Terminating    0     78m
oss-static-66fbb85b67-dqbl2  0/1   Terminating    0     78m
oss-static-66fbb85b67-zlvmw  1/1   Running      0     <invalid>
```

7. Run the following command to query the recreated pod:

```
kubectl get pod
```

Expected output:

```
NAME                READY  STATUS   RESTARTS  AGE
oss-static-66fbb85b67-zlvmw   1/1   Running  0     40s
```

8. Run the following command to verify that the *tmpfile* file exists in the */data* path. This indicates that data is persisted to the OSS volume.

```
kubectl exec oss-static-66fbb85b67-zlvmw ls /data | grep tmpfile
```

Expected output:

```
tmpfile
```

9.7. CPFS volumes

9.7.1. Use CPFS volumes in ACK clusters

You can use Cloud Paralleled File System (CPFS) volumes in Container Service for Kubernetes (ACK) clusters. This topic describes how to install the FlexVolume plug-in in ACK clusters and use CPFS volumes in pods.

Prerequisites

- A dedicated Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群.
- The ACK cluster can be accessed from the Internet through Secure Shell (SSH). For more information, see Use SSH to connect to an ACK cluster.

> **Notice** ACK supports only CPFS 1.0. CPFS 2.0 is not supported.

Context

CPFS is a parallel file system. CPFS stores data across multiple data nodes in a cluster and allows data to be simultaneously accessed by multiple clients. Therefore, CPFS can provide data storage services with high input/output operations per second (IOPS), high throughput, and low latency for large-sized and high-performance computing clusters. CPFS is a shared storage service that meets the requirements on resource sharing and high performance. We recommend that you use ACK with CPFS in scenarios such as big data, artificial intelligence (AI), and genetic computing. For more information about CPFS, see What is CPFS?.

Step 1: Install drivers

To use CPFS volumes in an ACK cluster, you must install the following drivers:

- CPFS container driver: the flexvolume-cpfs plug-in that is compatible with all CentOS versions. You can deploy flexvolume-cpfs to install the CPFS container driver.
- CPFS client driver: the driver of the CPFS client. It is similar to nfs-client. This driver is heavily reliant on the operating system kernel. You can install the CPFS client driver by using the following methods:
  - Manually install the driver. For more information, see Mount a file system.
  - When you deploy flexvolume-cpfs, the CPFS client driver is automatically installed. However, the driver does not support all operating system kernels.
    You can run the `uname -a` command on a node to query the kernel version of the operating system. You can install the CPFS client driver on the node that uses one of the following kernel versions:

```
3.10.0-957.5.1
3.10.0-957.21.3
3.10.0-1062.9.1
```

> **Note**
>
> - FlexVolume allows you to only install the CPFS client driver (cpfs-client). After you install the driver on a node, you cannot reinstall and upgrade the driver.
> - When you upgrade FlexVolume, only flexvolume-cpfs (CPFS container driver) is upgraded. cpfs-client (CPFS client driver) is not upgraded.
> - When you install flexvolume-cpfs on nodes where cpfs-client and lustre are deployed, cpfs-client of the latest version is not automatically installed.
> - You can only manually upgrade cpfs-client. For more information, see Mount a file system.

1. Deploy a YAML template on a node.
   i. Connect to Kubernetes clusters by using kubectl from a client.
   ii. Create a *flexvolume-cpfs.yaml* file.
   iii. Copy the following content to the file:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: flexvolume-cpfs
 namespace: kube-system
 labels:
  k8s-volume: flexvolume-cpfs
spec:
 selector:
  matchLabels:
   name: flexvolume-cpfs
 template:
  metadata:
   labels:
    name: flexvolume-cpfs
  spec:
   hostPID: true
   hostNetwork: true
   tolerations:
   - operator: "Exists"
   priorityClassName: system-node-critical
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
       - key: type
         operator: NotIn
         values:
         - virtual-kubelet
   containers:
   - name: flexvolume-cpfs
    image: registry.cn-hangzhou.aliyuncs.com/acs/flexvolume:v1.14.8.96-0d85fd1-aliyun
    imagePullPolicy: Always
    securityContext:
     privileged: true
    env:
    - name: ACS_CPFS
     value: "true"
```

```
      - name: FIX_ISSUES
        value: "false"
      livenessProbe:
        exec:
          command:
          - sh
          - -c
          - ls /acs/flexvolume
        failureThreshold: 8
        initialDelaySeconds: 15
        periodSeconds: 60
        successThreshold: 1
        timeoutSeconds: 15
      volumeMounts:
      - name: usrdir
        mountPath: /host/usr/
      - name: etcdir
        mountPath: /host/etc/
      - name: logdir
        mountPath: /var/log/alicloud/
      - mountPath: /var/lib/kubelet
        mountPropagation: Bidirectional
        name: kubeletdir
    volumes:
    - name: usrdir
      hostPath:
        path: /usr/
    - name: etcdir
      hostPath:
        path: /etc/
    - name: logdir
      hostPath:
        path: /var/log/alicloud/
    - hostPath:
        path: /var/lib/kubelet
        type: Directory
      name: kubeletdir
  updateStrategy:
    rollingUpdate:
      maxUnavailable: 10%
    type: RollingUpdate
```

iv. Run the following command to deploy the YAML file on the node:

```
kubectl create -f flexvolume-cpfs.yaml
```

2. Check the deployment result.

i. Run the following command to verify that the volume plug-ins are deployed:

```
kubectl get pod -nkube-system | grep flex
```

The following output is returned:

```
flexvolume-97psk                 1/1   Running  0      27m
flexvolume-cpfs-dgxfq              1/1   Running  0      98s
flexvolume-cpfs-qpbcb              1/1   Running  0      98s
flexvolume-cpfs-vlrf9              1/1   Running  0      98s
flexvolume-cpfs-wklls              1/1   Running  0      98s
flexvolume-cpfs-xtl9b              1/1   Running  0      98s
flexvolume-j8zjr                 1/1   Running  0      27m
flexvolume-pcg4l                 1/1   Running  0      27m
flexvolume-tjxxn                 1/1   Running  0      27m
flexvolume-x7ljw                 1/1   Running  0      27m
```

> **Note** The flexvolume-cpfs plug-in is deployed on pods whose names are prefixed with flexvolume-cpfs. Pods whose names do not contain cpfs are deployed with the FlexVolume plug-in. Alibaba Cloud disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets can be mounted to these pods. Both types of plug-ins can be deployed on the same node.

ii. Run the following command to check whether cpfs-client is installed:

```
rpm -qa | grep cpfs
```

The following output is returned:

```
kmod-cpfs-client-2.10.8-202.el7.x86_64
cpfs-client-2.10.8-202.el7.x86_64
```

iii. Run the following command to check whether mount.lustre is installed:

```
which mount.lustre
```

The following output is returned:

```
/usr/sbin/mount.lustre
```

Step 2: Use a CPFS volume

To use a CPFS volume in an ACK cluster, you must first create a CPFS file system and a mount target in the CPFS console. For more information, see Create a file system.

> **Notice** When you create a CPFS mount target, select a virtual private cloud (VPC). The ACK cluster and the mount target must be deployed in the same VPC.

In the following example, the mount target is *cpfs-*-alup.cn-shenzhen.cpfs.nas.aliyuncs.com@tcp:cpfs--ws5v.cn-shenzhen.cpfs.nas.aliyuncs.com@tcp* and the file system ID is *0237ef41*.

1. Create a persistent volume (PV).
   i. Create a *pv-cpfs.yaml* file.

ii. Copy the following content to the file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
 name: pv-cpfs
 labels:
  alicloud-pvname: pv-cpfs
spec:
 capacity:
  storage: 5Gi
 accessModes:
  - ReadWriteMany
 flexVolume:
  driver: "alicloud/cpfs"
  options:
   server: "cpfs-****-alup.cn-shenzhen.cpfs.nas.aliyuncs.com@tcp:cpfs-***-ws5v.cn-shenzhen.cpfs.nas.
aliyuncs.com@tcp"
   fileSystem: "0237ef41"
   subPath: "/k8s"
   options: "ro"
```

Parameter	Description
server	Set the value to the mount target of the CPFS file system.
fileSystem	Set the value to the ID of the CPFS file system.
subPath	Set the value to a subdirectory of the CPFS root.
options	Other mount options. This parameter is optional.

iii. Run the following command to create a PV:

```
kubectl create -f pv-cpfs.yaml
```

2. Create a persistent volume claim (PVC).

i. Create a *pvc-cpfs* file and copy the following content to the file:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: pvc-cpfs
spec:
 accessModes:
  - ReadWriteMany
 resources:
  requests:
   storage: 5Gi
 selector:
  matchLabels:
   alicloud-pvname: pv-cpfs
```

ii. Run the `kubectl create -f pvc-cpfs` command to create a PVC.

3. Create a Deployment.

i. Create a *nas-cpfs* file and copy the following content to the file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nas-cpfs
 labels:
  app: nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
     - name: pvc-cpfs
       mountPath: "/data"
   volumes:
    - name: pvc-cpfs
      persistentVolumeClaim:
       claimName: pvc-cpfs
```

ii. Run the `kubectl create -f nas-cpfs` command to create a Deployment.

Result

Run the following command to query the mounting states of the pods that run on the node:

```
kubectl get pod
```

The following output is returned:

```
NAME          READY  STATUS  RESTARTS  AGE
nas-cpfs-79964997f5-kzrtp  1/1   Running  0     45s
```

Run the following command to query the directories that are mounted to pods on the node:

```
kubectl exec -ti nas-cpfs-79964997f5-kzrtp sh
mount | grep k8s
```

The following output is returned:

```
192.168.1.12@tcp:192.168.1.10@tcp:/0237ef41/k8s on /data type lustre (ro,lazystatfs)
```

Run the following command to query the mounted directories on the node:

```
mount | grep cpfs
```

The following output is returned:

```
192.168.1.12@tcp:192.168.1.10@tcp:/0237ef41/k8s on /var/lib/kubelet/pods/c4684de2-26ce-11ea-abbd-00163e12e203/volumes/alicloud~cpfs/pv-cpfs type lustre (ro,lazystatfs)
```

Related information

- Use the CSI-CPFS plug-in

9.8. Create a PVC

In the Container Service for Kubernetes (ACK) console, you can create a persistent volume claim (PVC).

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- A persistent volume (PV) is created. In this example, a PV is created based on a cloud disk. For more information, see Usage notes for disk volumes.
  By default, PVCs are associated with PVs that have the alicloud-pvname label. This label is added to the PVs that are created in the ACK console. If a PV does not have this label, add the label to the PV before you can associate the PV with a PVC.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.
5. On the **Persistent Volumes Claims** tab, click **Create**.
6. In the **Create PVC** dialog box, set the parameters and click **Create**.
   - **PVC Type**: Cloud Disk, NAS, and OSS are supported. These options are also available when you set PVC Type for a PV.
   - **Name**: the name of the PVC.
   - **Allocation Mode**: **Use StorageClass**, **Existing Volumes**, and **Create Volume** are supported. In this example, **Use StorageClass** or **Existing Volumes** is selected.
   - **Existing Storage Class**: Click **Select PV**. In the Select PV dialog box, find the PV that you want to use and click **Select** in the Actions column of the PV.

     > **Note** This parameter is required only when you set Allocation Mode to **Use StorageClass**.

   - **Existing Volumes**: Click **Select PV**. In the Select PV dialog box, find the PV that you want to use and click **Select** in the Actions column of the PV.

     > **Note** This parameter is required only when you set Allocation Mode to **Existing Volumes**.

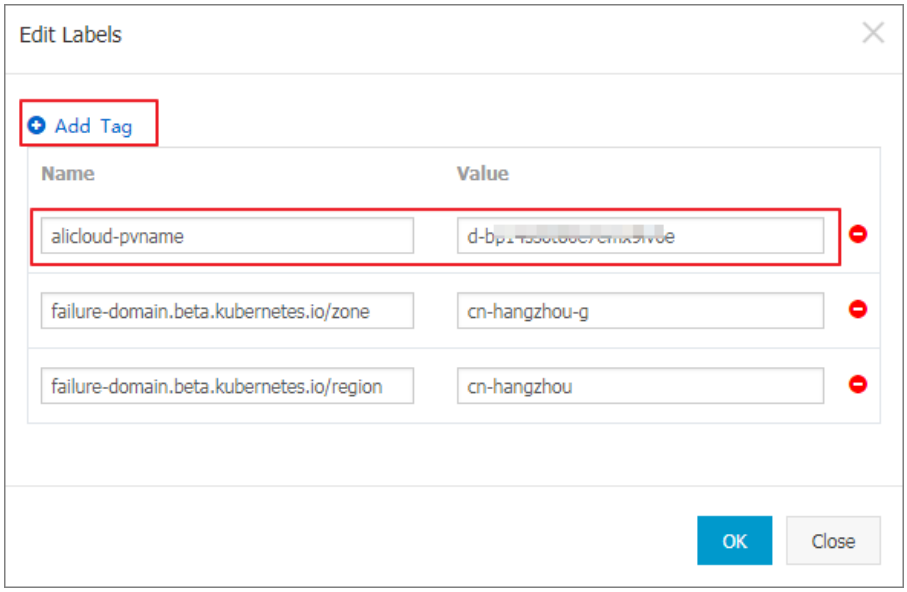
   - **Capacity**: the claimed usage. The value cannot be larger than the total capacity of the associated PV.

- **Access Mode**: Default value: ReadWriteOnce.

> **Note** This parameter is required only when you set Allocation Mode to **Use StorageClass**.

> **Note** Your cluster may have an unused PV that does not appear in the **Select PV** dialog box. The possible reason is that the alicloud-pvname label has not been added to the PV.

If you cannot find available PVs, go to the Cluster Information page. In the left-side navigation pane, click **Persistent Volumes**. The Persistent Volume Claims tab appears. Click the Persistent Volumes tab, find the PV that you want to use, and then click **Manage Labels** in the Actions column of the PV. You can add a label to the PV and set the label name to alicloud-pvname and the value to the PV name. If the PV is created from a disk, the disk ID is used as the PV name.



7. On the **Persistent Volume Claims** tab, the newly created PVC appears.

9.9. Use a PVC

In the Container Service for Kubernetes (ACK) console, you can create an application from an image or a template. When you create the application, you can specify a persistent volume claim (PVC) that the application uses to request physical storage. In this example, an application is created from an image. You can also create an application from a template and specify a PVC in the template. For more information, see Usage notes for disk volumes.

Prerequisites

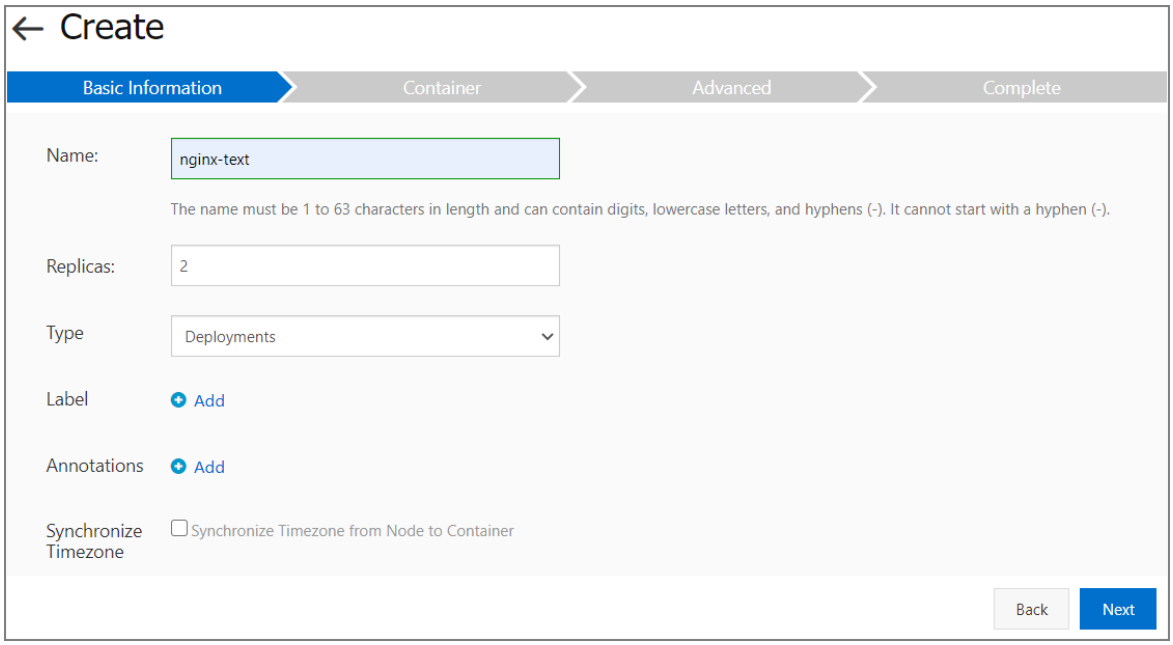
- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- A PVC is created. In this example, a PVC named pvc-disk is created based on cloud storage. For more information, see Create a PVC.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5.
6. On the **Basic Information** wizard page, configure the basic settings.

   For more information, see Configure basic settings.



7. On the **Container** wizard page, select an image and configure a data volume based on cloud storage. Cloud disks, NAS file systems, and Object Storage Service (OSS) buckets can be specified as cloud storage. In this example, select the pvc-disk PVC and click **Next**. For more information, see Configure the containers.
8. On the **Advanced** wizard page, create a service for the test-nginx application and click **Create**.
9. After the application is created, you are redirected to the **Creation Task Submitted** page. You can click **View Details** to view application details.

   The Basic Information page of the newly created test-nginx application appears by default.
10. On the **Pods** tab, find the pod to which the application belongs and click **View Details**.
11. You are redirected to the Overview page of the pod. Click the **Volumes** tab. Verify that the pod is associated with the pvc-disk PVC.

9.10. FAQ about the use of persistent volumes (PVs)

This topic provides answers to some commonly asked questions about the use of persistent volumes (PVs).

What can I do if a PV cannot be mounted?

Check whether FlexVolume is installed

Run the following command on a master node to obtain the pod information:

```
kubectl get pod -n kube-system | grep flexvolume
```

The following output is returned:

```
flexvolume-4wh8s      1/1    Running  0      8d
flexvolume-65z49      1/1    Running  0      8d
flexvolume-bpc6s      1/1    Running  0      8d
flexvolume-l8pml      1/1    Running  0      8d
flexvolume-mzkpv      1/1    Running  0      8d
flexvolume-wbfhv      1/1    Running  0      8d
flexvolume-xf5cs      1/1    Running  0      8d
```

Check whether the pod on which FlexVolume is installed is in the running state. Check whether the number of the running FlexVolume pods in a cluster is the same as the number of nodes that run these pods in the cluster.

If FlexVolume is not installed, install the plug-in. For more information, see Install and upgrade FlexVolume.

If the FlexVolume pods are not in the running state, troubleshoot the issue based on the logs of the FlexVolume plug-in.

Check whether the alicloud-disk-controller plug-in is installed

To use a cloud disk as a dynamically provisioned PV, you must install the alicloud-disk-controller plug-in. Run the following command to view the pod information:

```
kubectl get pod -n kube-system | grep alicloud-disk
```

The following output is returned:

```
alicloud-disk-controller-8679c9fc76-lq6zb   1/1 Running  0  7d
```

If the plug-in is not installed, install the plug-in. For more information, see Install and upgrade FlexVolume.

If a pod is not in the running state, troubleshoot the issue based on the logs of the plug-in.

How can I view storage logs?

Run the specified commands on Master node 1 to view FlexVolume logs

Run the following command to view the pod on which an error occurs:

```
kubectl get pod -n kube-system | grep flexvolume
```

Run the following command to view the logs of the pod that has an error:

```
kubectl logs flexvolume-4wh8s -n kube-system
kubectl describe pod flexvolume-4wh8s -n kube-system
```

> **Note** The last several entries of the returned pod information indicate the pod state. You can analyze the error based on the response.

View the logs of the cloud disk, NAS, and Object Storage Service (OSS) drivers.

Run the following command to view the PV logs on the host node: If a PV failed to be mounted to a pod, check the address of the node where the pod is located.

```
kubectl describe pod nginx-97dc96f7b-xbx8t | grep Node
```

The following output is returned:

```
Node: cn-hangzhou.i-bp19myla3uvnt6zihejb/192.168.XX.XX
Node-Selectors: <none>
```

Log on to the node and view the logs of the cloud disk, NAS, and OSS drivers.

```
ssh 192.168.XX.XX
ls /var/log/alicloud/flexvolume*
```

The following output is returned:

```
flexvolume_disk.log flexvolume_nas.log flexvolume_o#ss.log
```

Run the specified commands on Master node 1 to view the logs of the provisioner plug-in

Run the following command to view the pod on which an error occurs:

```
kubectl get pod -n kube-system | grep alicloud-disk
```

Run the following command to view the logs of the pod that has an error:

```
kubectl logs alicloud-disk-controller-8679c9fc76-lq6zb -n kube-system
kubectl describe pod alicloud-disk-controller-8679c9fc76-lq6zb -n kube-system
```

> **Note** The last several entries of the returned pod information indicate the pod state. You can analyze the error based on the response.

View Kubelet logs

If a PV failed to be mounted to a pod, check the address of the node where the pod is located.

```
kubectl describe pod nginx-97dc96f7b-xbx8t | grep Node
```

The following output is returned:

```
Node: cn-hangzhou.i-bp19myla3uvnt6zihejb/192.168.XX.XX
Node-Selectors: <none>
```

Log on to the node to view kubelet logs.

```
ssh 192.168.XX.XX
journalctl -u kubelet -r -n 1000 &> kubelet.log
```

> **Note** In the preceding command, -n specifies the required number of log entries to return.

The preceding steps describe how to obtain the error logs of the FlexVolume, provisioner, and Kubelet plug-ins. If the issue cannot be fixed based on the logs, contact the technical support team of Alibaba Cloud and provide related log information.

FAQ of cloud disks

What can I do if a timeout error has occurred when I mounted a PV to a pod?

If the node that runs the pod is manually added, this issue may be caused by insufficient Security Token Service (STS) permissions. We recommend that you manually grant the RAM permissions. For more information, see Bind an instance RAM role.

What can I do if a size error has occurred when I mounted a PV to a pod?

To create a cloud disk, the disk size must meet the following requirements:

> **Note**
>
> - A basic disk must have a minimum capacity of 5 GiB.
> - An ultra disk must have a minimum capacity of 20 GiB.
> - A standard SSD must have a minimum capacity of 20 GiB.

What can I do if a zone error has occurred when I mounted a PV to a pod?

To mount a cloud disk to an Elastic Compute Service (ECS) instance, the cloud disk and the ECS instance must be deployed in the same region and zone.

What can I do if an input/output error has occurred in a cloud disk after a system upgrade?

1. Upgrade FlexVolume to v1.9.7-42e8198 or later.
2. Recreate the pod that has the error.

Run the following command to upgrade FlexVolume:

```
kubectl set image daemonset/flexvolume acs-flexvolume=registry.cn-hangzhou.aliyuncs.com/acs/flexvolume:v1.9.7-42e8198 -n kube-system
```

To obtain the latest FlexVolume version, log on to the Container Registry console. In the left-side navigation pane, choose Image Hub > **Search**. On the Search page, select User Public Image from the drop-down list and enter acs/flexvolume in the search bar.

FAQ about NAS

What can I do if it took a long time to mount a NAS file system to pod?

If the NAS file system contains a large amount of data and you set the chmod parameter in the mounting template, this issue may occur. We recommend that you remove the chmod parameter.

What can I do if a timeout error has occurred when I mounted a NAS file system to a pod?

Make sure that the mount target of the NAS file system and the cluster are deployed in the same virtual private cloud (VPC).

FAQ of OSS

What can I do if an OSS bucket failed to be mounted?

- Check whether the AccessKey information is invalid.
- Check whether the URL used to mount the OSS bucket is accessible over the network.

What can I do if the OSS bucket directory that is mounted to a container is unavailable after an upgrade of the cluster that runs the pod?

If you upgrade your Container Service for Kubernetes (ACK) cluster or restart a kubelet, the container network is restarted. This causes the OSSFS process to restart.

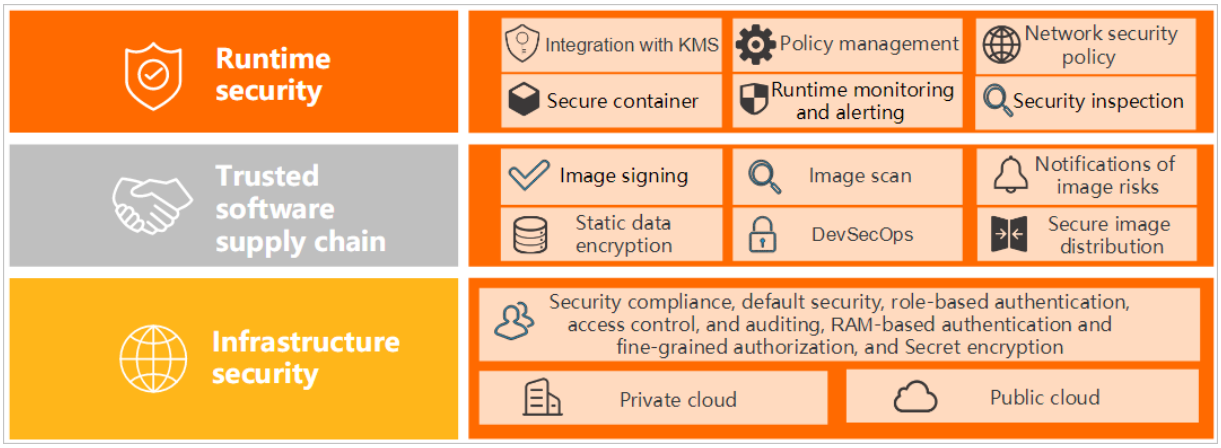
After the OSSFS process restarts, the mapping between the host and container directory becomes invalid. In this case, restart the container or recreate the pod. You can configure the health check feature to automatically restart a container or pod.

For more information about OSS, see Mount OSS volumes.

10.Security management

10.1. Security system overview

This topic describes the security system of Container Service for Kubernetes (ACK) built on top of runtime security, trusted software supply chains, and infrastructure security. This security system is supported by a variety of features provided by ACK , including security inspection, policy management, runtime monitoring and alerting, image scans, image signing, cloud-native application delivery chain, default security, identity management, and fine-grained access control.



Runtime security

- Security inspection
  Developers must follow the least privilege principle when they configure pod templates for deployments. Otherwise, attackers can exploit unnecessary permissions that are granted to users on pods to launch escape attacks against containers. ACK supports security inspections for runtimes. This feature inspects pod configurations for potential risks in real time.
  An inspection report is generated after a security inspection is performed. The report includes the description of each inspection item and suggestions about how to fix the related security issues. You can also configure periodic inspections. The results of each periodic inspection are logged to the related Logstore in Log Service. For more information, see Use the inspection feature to check for security risks in the workloads of an ACK cluster.
- Policy management
  Pod security policy is a significant method to verify the security of pod configurations before pods are deployed. This ensures that applications are running in secure pods. You can use pod security policies to enforce security verification on pods where applications will be deployed. Pod security policies function as a scan engine, such as AppArmor and SELinux, to provide in-depth protection for clusters.
  A pod security policy is a cluster-level Kubernetes-native resource that is enforced by the admission control mechanism of the Kubernetes API server. After you enable pod security policy control, security verification is enforced on the configurations of pods to be created. If a pod fails to meet the conditions that are defined in a specified pod security policy, the Kubernetes API server rejects the request to create the pod.
  By default, pod security policy control is enabled for ACK clusters. You can enable or disable the admission controller of pod security policy in the ACK console. You can also customize pod security policies or bind a pod security policy to a specified service account in the ACK console. This makes pod security policies mush easier to use and avoids issues where existing applications cannot be redeployed after they are bound to pod security policies. For more information, see Use a PSP.
- Runtime monitoring and alerting

Cloud-native applications are deployed in containers after they pass the authentication, authorization, and admission control of the API server. However, based on the zero trust principle for application security, monitoring and alerting are required to ensure the security of application runtimes. Therefore, ACK is deeply integrated with the alerting and vulnerability detection capabilities of Security Center. This allows cluster administrators to monitor application runtimes and raise alerts upon security events. Runtime monitoring and alerting are used to prevent the following attacks that are launched against containers:

- Loading of malicious images
- Implanting of viruses and malicious programs
- Intrusion into containers
- Container escapes and high-risk operations

On the details page of an ACK cluster, choose **Security > Runtime Security** to view the received alerts in real time. You can also follow the instructions on the page to check and handle alerts. For more information, see Use Runtime Security to monitor ACK clusters and configure alerts.

- Sandboxed-Container
  Sandboxed-Container is an alternative to the Docker runtime. It allows you to run applications in a sandboxed and lightweight virtual machine that has a dedicated kernel. This enhances resource isolation and improves security.
  Sandboxed-Container is suitable in scenarios such as untrusted application isolation, fault isolation, performance isolation, and load isolation among multiple users. Sandboxed-Container provides higher security. Sandboxed-Container has minor impacts on application performance and offers the same user experience as Docker in terms of logging, monitoring, and elastic scaling. For more information about Sandboxed-Container, see Sandboxed-Container overview.
- TEE-based confidential computing
  Container Service for Kubernetes provides trusted execution environment (TEE)-based confidential computing. This is a cloud-native and all-in-one solution that uses hardware encryption technologies. TEE-based confidential computing ensures the security, integrity, and confidentiality of data. It also simplifies the development and release of trusted and confidential applications and reduces management costs. Therefore, TEE-based confidential computing is suitable for users in the financial industry and government who require high security.
  TEE-based confidential computing allows you to store sensitive data and code in a TEE. This prevents the other parts of the system from accessing the data and code. Data and code in a TEE are inaccessible to other applications, the BIOS, the operating system, the kernel, administrators, O&M personnel, cloud service providers, or hardware components except CPUs. This simplifies data management and reduces the risk of sensitive data leakage. For more information, see TEE-based confidential computing.

Trusted software supply chains

- Image scans
  Container Registry allows you to scan all container images that use Linux for known vulnerabilities. After you run a scan, you will receive a report that contains information about the detected vulnerabilities and suggestions on how to fix them. Image scans help you reduce the risks in container images. Container Registry is also integrated with the scan engine of Security Center. This engine can be used to detect system vulnerabilities, application vulnerabilities, and malicious samples in images. For more information about image scans, see Scan container images.
- Image signing

When you manage container images, you can use the content trust mechanism to verify whether the images and their publishers are trusted. Image publishers can use digital signatures to sign images. The digital signatures are stored in Container Registry. Then, you can verify the signatures of images to ensure that only images signed by trusted authorities are deployed. This reduces the risk of malicious code execution and ensures the security and traceability of container images from the software supply chain to application deployment. For more information about how to sign an image and verify the signature, see Use kritis-validation-hook to automatically verify the signatures of container images.

- Cloud-native application delivery chain
  Container Registry provides a cloud-native application delivery chain for you to develop containerized applications with high security and efficiency. This chain covers image builds, image scans, image synchronization on a global scale, and image deployment. You can also customize fine-grained security policies. The cloud-native application delivery chain makes the entire lifecycle of application development secure, observable, and traceable. After you use the cloud-native application delivery chain, you only need to submit the code once for each application. The image is distributed and deployed across all regions worldwide in a secure and efficient manner. This upgrades the development pipeline from DevOps to DevSecOps. For more information about the cloud-native application delivery chain, see Create a delivery chain.

Infrastructure security

- Default security
  In an ACK cluster, the security of nodes and components on the control plane is reinforced based on Center for Internet Security (CIS) Kubernetes Benchmark. In addition, the configurations of all system components are reinforced by following the ACK best practices for security. No component image contains critical vulnerabilities that are identified by Common Vulnerabilities and Exposures (CVE).
  - Each newly created ACK cluster is assigned a security group that allows only inbound Internet Control Message Protocol (ICMP) packets sent from the Internet. By default, you cannot connect to an ACK cluster by using SSH over the Internet. If you want to connect to an ACK cluster by using SSH over the Internet, see Use SSH to connect to an ACK cluster.
  - You can enable Internet access for nodes in an ACK cluster by using NAT gateways. This secures Internet access and reduces security risks.
  - Worker nodes in a managed Kubernetes cluster are assigned Resource Access Management (RAM) roles that are authorized by following the least privilege principle. These RAM roles have only the minimum permissions on Alibaba Cloud services. For more information, see ACK reduces the permissions of worker RAM roles in managed Kubernetes clusters.
- Identity management
  The communication and data transmission among all components in an ACK cluster must be secured by using TLS-based authentication. In addition, ACK automatically renews the certificates of system components. The kubeconfig file contains credentials that are required when you connect to the API server of a cluster. You can log on to the ACK console or call the ACK API as a RAM user or by assuming a RAM role and then obtain the kubeconfig file. For more information, see DescribeClusterUserKubeconfig. The cluster credentials are maintained by ACK. If the cluster credentials in the returned kubeconfig file are leaked, you must immediately revoke the kubeconfig file. For more information, see Revoke a KubeConfig credential.
  When you create an ACK cluster, you can enable **service account token volume projection**. This feature enhances the security when you use service accounts in applications. For more information, see Enable service account token volume projection.
- Fine-grained access control
  Based on role-based access control (RBAC), ACK provides fine-grained access control on Kubernetes resources in an ACK cluster. This is a basic but essential reinforcement for application security. On the **Authorizations** page in the ACK console, you can assign RBAC roles to grant fine-grained permissions that are scoped to namespaces. This authorization method provides the following benefits:

- ACK provides the following predefined RBAC roles: administrator, O&M engineer, developer, and restricted user. This provides an easy way to grant permissions to employees in hierarchical departments of an enterprise.
- ACK allows you to authorize multiple clusters or RAM users at a time.
- ACK allows you to authorize a RAM user that can be assumed by a RAM role.
- ACK allows you to assign custom cluster roles.

For more information, see Assign an RBAC role to a RAM user.
You can install the gatekeeper add-on on the Add-ons page in the ACK console. This add-on provides fine-grained access control by using the Open Policy Agent (OPA) policy engine. For more information, see gatekeeper.

- Auditing
  ACK is deeply integrated with Log Service. ACK can collect, retrieve, and visualize the following types of audit logs:
  - The audit log of the API server of a cluster. This type of audit log records the operations that are performed by users when they access the cluster. You can check the audit log to trace each operation if required. This is a key component to maintain clusters in a secured way. On the **Cluster Auditing** page, you can view a variety of auditing reports and also configure alerts for operations that are performed on specified resource types based on the log content. For more information, see Enable cluster auditing.
  - The audit log of Ingress traffic. Multiple visualized traffic reports are provided to show the status of Ingresses in a cluster. The reports provide information such as the page views (PVs) and unique visitors (UVs) of services, the ratio of request successes or failures, and the latency. Exceptions can also be automatically detected by using the machine learning algorithms provided by Log Service and the time series analysis algorithms. For more information, see Monitor nginx-ingress and analyze the access log of nginx-ingress.
  - The audit log of event monitoring. Event monitoring records cluster events in the audit log. You can diagnose anomalies and security risks in a cluster based on these events. For more information, see Event monitoring.

- Secret encryption
  Kubernetes Secrets are encoded in Base64 when they are stored in etcd. To further reinforce the security of the stored Kubernetes Secrets, you can use keys that are created in Key Management Service (KMS) to encrypt Secrets of professional managed Kubernetes clusters. For more information, see Use KMS to encrypt Kubernetes secrets at rest in the etcd.

10.2. Infrastructure security

10.2.1. Access applications in an ACK cluster through HTTPS

You can access applications in a Container Service for Kubernetes (ACK) cluster by using multiple methods. For example, you can request `<SLB-Instance-IP>:<Port>`, `<NodeIP>:<NodePort>`, or the domain name of the application. By default, you cannot access a Kubernetes cluster through HTTPS. You can enable secure HTTPS access by using ACK and Server Load Balancer (SLB). This topic describes how to enable HTTPS access to an ACK cluster by using a certificate.

Prerequisites

- 创建Kubernetes托管版集群.
- A server certificate is created for the cluster. The server certificate includes a public key certificate and a private key.

- You can run the following command to create a server certificate:

```
openssl genrsa -out tls.key 2048
```

The following output is returned:

```
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
```

```
ls
```

The following output is returned:

```
You are about to be asked to enter information that will be incorporated
...
-----
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:zhejiang
Locality Name (eg, city) [Default City]:hangzhou
Organization Name (eg, company) [Default Company Ltd]:alibaba
Organizational Unit Name (eg, section) []:test
Common Name (eg, your name or your server's hostname) []:foo.bar.com       # You must specify a valid dom
ain name.
Email Address []:a@alibaba.com
```

- You can also purchase a server certificate that is issued by Alibaba Cloud. For more information, see Use a certificate from Alibaba Cloud SSL Certificates Service.

Context

You can select the following methods to configure the certificate based on how you want to access the cluster.

- Specify the certificate information in a frontend SLB instance.
- Specify the certificate information in an Ingress.

Specify the certificate information in a frontend SLB instance

If you use this method, take note of the following items:

- Advantage: A frontend SLB instance is configured with the certificate information to accept external access. Access from within the cluster is still based on HTTP.
- Disadvantage: You must maintain the mappings between a large number of domain names and IP addresses.
- Scenario: Your application is accessed through LoadBalancer type Services rather than Ingresses.

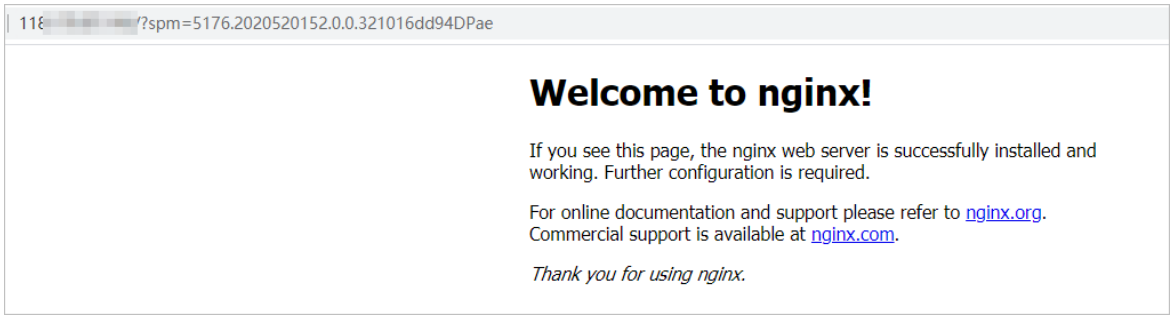
Preparations

An NGINX application is deployed in the cluster. You can access the application through a LoadBalancer type Service. For more information, see Create a stateless application by using a Deployment.

Examples:

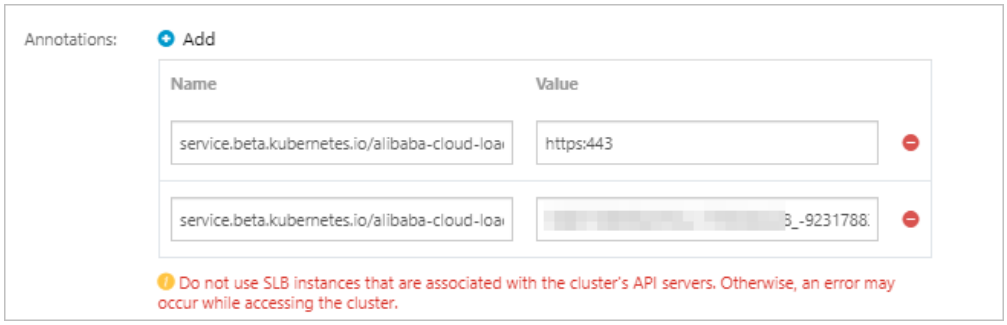
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

4. In the left-side navigation pane of the details page, choose **Network > Services**.
5. Select the namespace where the related Service is deployed and click the external endpoint to access the application. The endpoint is in the `<SLB IP>:<Port>` format.



6. Log on to the SLB console.
7. Configure an **SSL Certificate**.
   - If you have created a server certificate by running commands as described in the prerequisites, you need to upload the created certificate, including the public key certificate and the private key, to Alibaba Cloud. For more information, see Upload a third-party certificate.
   - If you have purchased a server certificate that is issued by Alibaba Cloud, skip this step. For more information about how to purchase a server certificate that is issued by Alibaba Cloud, see Use a certificate from Alibaba Cloud SSL Certificates Service.

     Find the certificate that you want to use from the certificate list.
8. On the Services page of the ACK console, find the created Service and click **Update** in the **Actions** column.
9. In the **Update Service** dialog box, enter the annotations as shown in the following figure.



Annotation	Name	Value
Annotation 1	service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port	https:443
Annotation 2	service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id	${YOUR_CERT_ID}

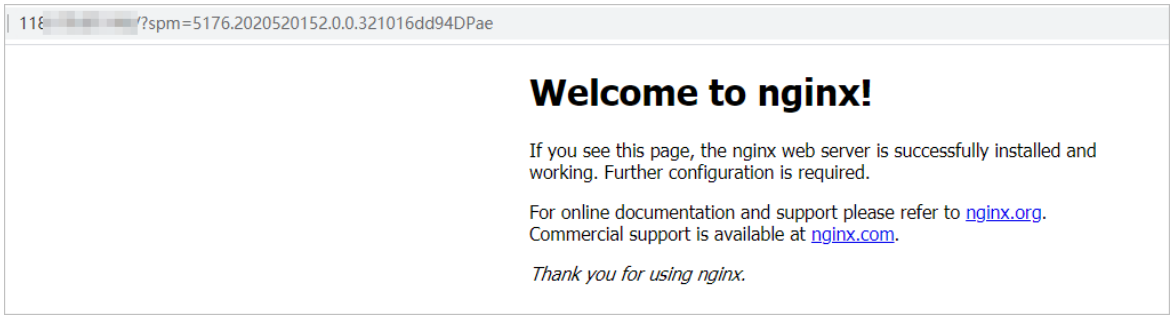
> **Note** Replace ${YOUR_CERT_ID} with the ID of the certificate that is configured in Step 7.

You can also add the annotations by using YAML files. The following YAML file is an example:

```
apiVersion: v1
kind: Service
metadata:
 annotations:
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-protocol-port: "https:443"
  service.beta.kubernetes.io/alibaba-cloud-loadbalancer-cert-id: "${YOUR_CERT_ID}"
 name: nginx
 namespace: default
spec:
 ports:
 - name: https
  port: 443
  protocol: TCP
  targetPort: 80
 - name: http
  port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: nginx
 type: LoadBalancer
```

Note Set targetPort to 80. This way, requests to HTTPS port 443 are redirected to HTTP port 80.

10. Enter `https://<slb-instance-ip>` into the address bar of your browser to access the NGINX application through HTTPS.



Specify the certificate information in an Ingress

If you use this method, take note of the following items:

- Advantage: You do not need to modify the SLB configurations. You can separately manage the certificate of each application by using Ingresses.
- Scenario: Each application in your cluster requires a separate certificate, or an application in the cluster can be accessed only by using a certificate.

Preparations

A Tomcat application is created in your cluster. You can access the application through a ClusterIP type Service. In this example, an Ingress is used to enable external access through HTTPS. For more information, see Create a stateless application by using a Deployment.

Examples:

1. Run the following command to create a Secret by using the certificate that is created in the prerequisites:

> **Note** You must specify a valid domain name. Otherwise, an error occurs when you access the application through HTTPS.

```
kubectl create secret tls secret-https --key tls.key --cert tls.crt
```

2. Log on to the ACK console.
3. In the left-side navigation pane of the ACK console, click **Clusters**.
4. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
5. In the left-side navigation pane of the details page, click **Namespaces and Quotas**. In the upper-right corner of the Namespace page, click **Create**.
6. (Optional)In the dialog box that appears, configure an Ingress that can be accessed through HTTPS and click **Create**.

   For more information, see Create an Ingress. In this example, set the following parameters:

   - **Name**: Enter a name for the Ingress.
   - **Domain**: Enter the domain name that is specified in the preceding steps. This domain name must be the same as the domain name that is specified in the SSL certificate.
   - **Service**: Select the Service that is related to the Tomcat application. The port number is 8080.
   - **EnableTLS**: After you select EnableTLS, select the created Secret.

   You can also create an Ingress by using YAML files. The following YAML template is an example:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tomcat-https
spec:
 tls:
 - hosts:
  - foo.bar.com
  secretName: secret-https
 rules:
 - host: foo.bar.com
  http:
   paths:
   - path: /
     backend:
       serviceName: tomcat-svc
       servicePort: 8080
```

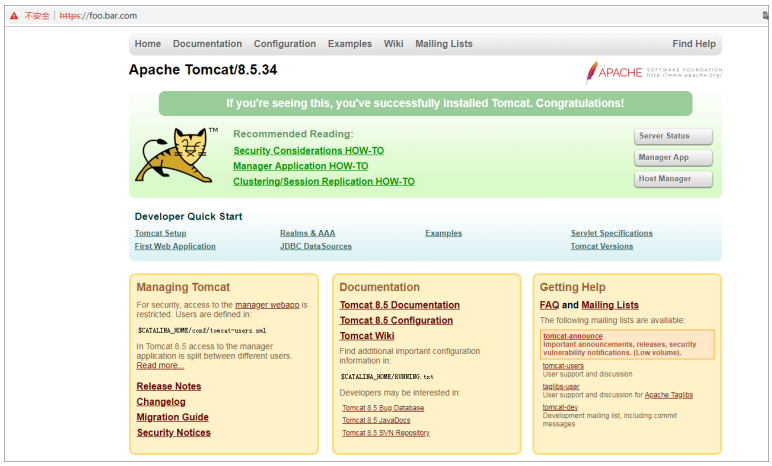
7. Return to the Ingresses page to view the newly created Ingress. The endpoint and domain name of the Ingress also appear on the page. In this example, the domain name is `foo.bar.com`. You can also view these details on the details page of the Ingress.

   > **Note** In this example, `foo.bar.com` is used as a test domain name. You must add a mapping to the hosts file.

   ```
   47.110.119.203 foo.bar.com          # The IP address is the endpoint of the Ingress.
   ```

8. Enter `https://foo.bar.com` into the address bar of your browser to access the Tomcat application.

> **Note** A TLS certificate is created and configured for the Ingress. Therefore, you must access the domain name through HTTPS. In this example, `foo.bar.com` is resolved on a local DNS server. You must use a domain name that has obtained an Internet Content Provider (ICP) number.



10.2.2. Enable cluster auditing

In a Kubernetes cluster, kube-apiserver collects audit log data that helps administrators track operations performed by different users. This plays an essential role in security and maintenance of the cluster. This topic describes how to configure parameters for cluster auditing, how to enable Log Service to collect and analyze audit log data, how to set custom alert rules, and how to disable cluster auditing.

Configure parameters for cluster auditing

By default, **Enable Log Service** is selected when you create a cluster. This indicates that kube-apiserver automatically collects audit log data for the cluster. The following table describes the parameters of cluster auditing.

> **Note** Log on to a master node. You can find the configuration file of kube-apiserver in the following path: */etc/kubernetes/manifests/kube-apiserver.yaml*.

Parameter	Description
audit-log-maxbackup	A maximum of 10 audit log files can be retained.
audit-log-maxsize	The maximum size of an audit log file is 100 MB.
audit-log-path	The log files are stored in the */var/log/kubernetes/kubernetes.audit* path.
audit-log-maxage	Audit log files are retained for a maximum of seven days.
audit-policy-file	The path of the audit policy file is */etc/kubernetes/audit-policy.yml*.

Log on to a master node. You can find the audit policy file in the following path: */etc/kubernetes/audit-policy.yml*. The audit policy file contains the following content:

```
apiVersion: audit.k8s.io/v1beta1 # This is required.
kind: Policy
```

```
# Don't generate audit events for all requests in RequestReceived stage.
omitStages:
 - "RequestReceived"
rules:
 # The following requests were manually identified as high-volume and low-risk,
 # so drop them.
 - level: None
  users: ["system:kube-proxy"]
  verbs: ["watch"]
  resources:
   - group: "" # core
    resources: ["endpoints", "services"]
 - level: None
  users: ["system:unsecured"]
  namespaces: ["kube-system"]
  verbs: ["get"]
  resources:
   - group: "" # core
    resources: ["configmaps"]
 - level: None
  users: ["kubelet"] # legacy kubelet identity
  verbs: ["get"]
  resources:
   - group: "" # core
    resources: ["nodes"]
 - level: None
  userGroups: ["system:nodes"]
  verbs: ["get"]
  resources:
   - group: "" # core
    resources: ["nodes"]
 - level: None
  users:
   - system:kube-controller-manager
   - system:kube-scheduler
   - system:serviceaccount:kube-system:endpoint-controller
  verbs: ["get", "update"]
  namespaces: ["kube-system"]
  resources:
   - group: "" # core
    resources: ["endpoints"]
 - level: None
  users: ["system:apiserver"]
  verbs: ["get"]
  resources:
   - group: "" # core
    resources: ["namespaces"]
 # Don't log these read-only URLs.
 - level: None
  nonResourceURLs:
   - /healthz*
   - /version
   - /swagger*
 # Don't log events requests.
 - level: None
  resources:
   - group: "" # core
```

```
    resources: ["events"]
# Secrets, ConfigMaps, and TokenReviews can contain sensitive & binary data,
# so only log at the Metadata level.
- level: Metadata
 resources:
  - group: "" # core
    resources: ["secrets", "configmaps"]
  - group: authentication.k8s.io
    resources: ["tokenreviews"]
# Get repsonses can be large; skip them.
- level: Request
 verbs: ["get", "list", "watch"]
 resources:
  - group: "" # core
  - group: "admissionregistration.k8s.io"
  - group: "apps"
  - group: "authentication.k8s.io"
  - group: "authorization.k8s.io"
  - group: "autoscaling"
  - group: "batch"
  - group: "certificates.k8s.io"
  - group: "extensions"
  - group: "networking.k8s.io"
  - group: "policy"
  - group: "rbac.authorization.k8s.io"
  - group: "settings.k8s.io"
  - group: "storage.k8s.io"
# Default level for known APIs
- level: RequestResponse
 resources:
  - group: "" # core
  - group: "admissionregistration.k8s.io"
  - group: "apps"
  - group: "authentication.k8s.io"
  - group: "authorization.k8s.io"
  - group: "autoscaling"
  - group: "batch"
  - group: "certificates.k8s.io"
  - group: "extensions"
  - group: "networking.k8s.io"
  - group: "policy"
  - group: "rbac.authorization.k8s.io"
  - group: "settings.k8s.io"
  - group: "storage.k8s.io"
# Default level for all other requests.
- level: Metadata
```

> **Note**
>
> - Requests are not logged upon reception. Requests are logged only after response headers are sent.
> - The following types of requests are not logged: watch requests by kube-proxy, GET requests by kubelet and *system:nodes* for node resources, operations on endpoint resources by kube components in the kube-system namespace, and GET requests by kube-apiserver for namespace resources.
> - Requests with read-only URLs that match */healthz**, */version**, or */swagger** are not logged.
> - Requests for Secrets, ConfigMaps, and TokenReview resources are logged at the metadata level because these resources may contain sensitive information or binary files. Cluster auditing at the metadata level only collects request metadata, such as request users, timestamps, request resources, and actions. The information about the request body or the response body is not logged.
> - Requests for sensitive resources such as authentication, role-based access control (RBAC), certificates, auto scaling, and storage resources are logged, including the request body and the response body.

View audit log reports

Container Service for Kubernetes (ACK) provides three audit log reports for each cluster. You can find the following information in these reports:

- Important operations performed by users and system components on the cluster.
- The source IP addresses of these operations and the regional distribution of these IP addresses.
- The details of operations on each type of resource.
- The details of operations performed by Resource Access Management (RAM) users.
- Details of important operations, such as container logons, Secret retrieval, and resource deletion.

> **Note**
>
> - By default, the Enable Log Service check box is selected when you create a cluster. This indicates that cluster auditing is automatically enabled. For more information about the billing methods of Log Service, see Billing rules. If Log Service is not activated, see Enable cluster auditing.
> - Do not modify audit log reports. If you want to customize reports, log on to the Log Service console to create new reports.

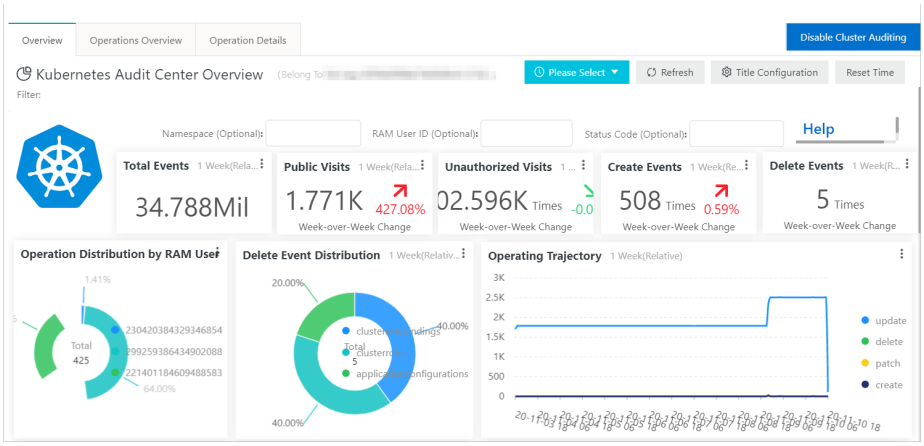
You can use one of the following methods to view audit log reports:

- Log on to the ACK console. In the left-side navigation pane, click **Clusters**. On the Clusters page, find the cluster that you want to view, and choose **More > Cluster Auditing** in the **Actions** column.
- Log on to the ACK console. In the left-side navigation pane, click **Clusters**. On the Clusters page, click the name of the cluster that you want to view. In the left-side navigation pane of the cluster details page, choose **Security > Cluster Auditing**.

Overview of audit log reports

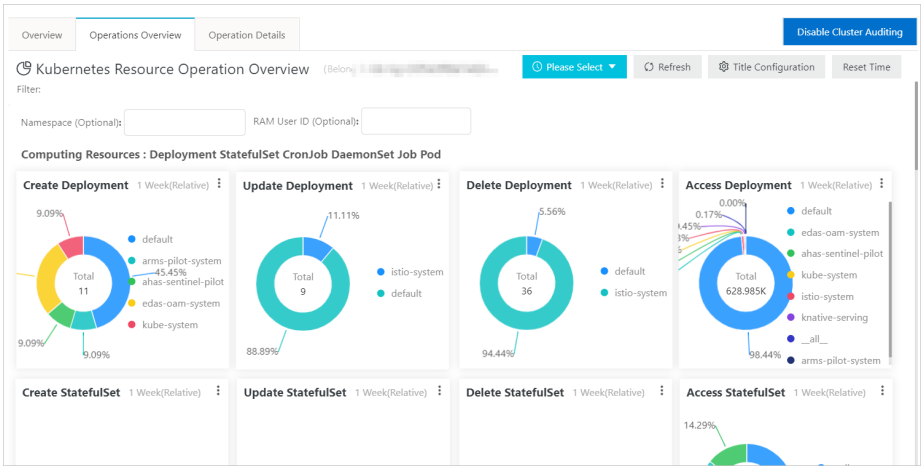
The Cluster Auditing page displays audit log reports on three tabs: Overview, Operations Overview, and Operation Details.

- Overview
  This report provides an overview of the events in the cluster and the details of important events, such as requests from the Internet, command executions, resource deletion, and Secret retrieval.



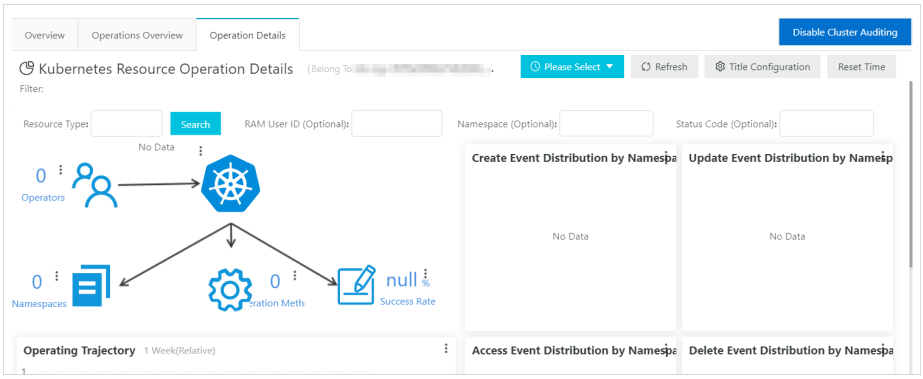
> **Note** By default, the report displays statistics of the last seven days. You can specify a time period and view the statistics collected during the period. You can filter the statistics by namespace, RAM user ID, and status code. You can also select one or more items to filter the statistics.

- Operations Overview
  This report provides statistics about common operations on computing resources, network resources, and storage resources in the cluster. The operations include creating resources, updating resources, deleting resources, and accessing resources. The following information is displayed:
  - Computing resources include Deployment, StatefulSet, CronJob, DaemonSet, Job, and pod.
  - Network resources include Service and Ingress.
  - Storage resources include ConfigMap, Secret, and Persistent Volume Claim.



> **Note**
>
> - By default, the report displays statistics of the last seven days. You can specify a time period and view the statistics collected during the period. You can filter the statistics by namespace and RAM user ID. You can also select one or more items to filter the statistics.
> - To view details of the operations on a resource, go to the Operation Details report.

- Operation Details
  This report provides operation details on a specific resource type. You can specify a resource type to query operation details in real time. The report contains the total number of operations, distribution of namespaces, operation success rate, temporal order of operations, and other operation details.



> **Note**
>
> - To query operations about a CustomResourceDefinition (CRD) resource registered in Kubernetes or a resource that is not listed in the report, enter the plural form of the resource name. For example, to query operations about a CRD resource named AliyunLogConfig, enter AliyunLogConfigs.
> - By default, the report displays statistics of the last seven days. You can specify a time period and view the statistics collected during the period. You can filter the statistics by namespace, RAM user ID, and status code. You can also select one or more items to filter the statistics.

View detailed log data

To customize queries or analyze audit log data, log on to the Log Service console and view detailed log data.

> **Note** By default, Log Service retains audit log files in a Logstore for 30 days. For information about how to change the log retention period, see Manage a Logstore.

1. Log on to the Log Service console.
2. Find the project of the cluster and click the project name.
3. Find Logstore audit-${clustered}, click the icon, and then select **Search & Analysis** to view the audit log data.

   > **Note**
   >
   > - During the cluster creation process, a Logstore named `audit-${clustereid}` is automatically created under the project.
   > - By default, indexes are set up for the Logstore. Do not modify the indexes.

You can query audit log data by using the following methods:

- To query the operations performed by a RAM user, enter the RAM user ID, and click **Search & Analyze**.
- To query the operations performed on a resource, enter the resource name, and click **Search & Analyze**.
- To filter out the operations performed by system components, enter `NOT user.username: node NOT user.username: serviceaccount NOT user.username: apiserver NOT user.username: kube-scheduler NOT user.username: kube-controller-manager`, and click **Search & Analyze**.

For more information about how to query log data, see Query methods.

Configure an alert rule

Log Service enables you to configure alert rules to monitor the operations that are performed on specific resources in real time. Available notification methods includeDingTalk chatbot webhooks, custom webhooks, and Alibaba Cloud Message Center. For more information, see Configure an alert rule for Log Service.

> **Note** For more information about how to query audit log data, see the query statements in audit reports. You can perform the following steps to view query statements: On the project details page, click the **Dashboard** icon in the left-side navigation pane. Select a dashboard to go to the details page. Select a chart, click the More icon, and then click **View Analysis Details**.

Example 1: Alerts upon command execution on containers

To monitor command execution on containers, alerts must be sent at the earliest opportunity when a user attempts to log on to a container or run commands on a container. The alert notification must include the following information: the container to which the user logs on, the commands, the user name, the event ID, the operation time, and the source IP address.

- Sample query statement:

```
verb : create and objectRef.subresource:exec and stage:  ResponseStarted | SELECT auditID as "Event ID", date_
format(from_unixtime(__time__), '%Y-%m-%d %T' ) as "Operation time", regexp_extract("requestURI", '([^\?]*
)/exec\?.*', 1)as "Resource",  regexp_extract("requestURI", '\?(.*)', 1)as "Command" ,"responseStatus.code" as "
Status code",
 CASE
 WHEN "user.username" != 'kubernetes-admin' then "user.username"
 WHEN "user.username" = 'kubernetes-admin' and regexp_like("annotations.authorization.k8s.io/reason", 'Rol
eBinding') then regexp_extract("annotations.authorization.k8s.io/reason", ' to User "(\w+)"', 1)
 ELSE 'kubernetes-admin' END
 as "Username",
CASE WHEN json_array_length(sourceIPs) = 1 then json_format(json_array_get(sourceIPs, 0)) ELSE  sourceIPs E
ND
as "Source IP address" limit 100
```

- The conditional expression is `Event =~ ".*"` .

Example 2: Alerts upon failed Internet connection requests to the API server

To protect a cluster against attacks launched from the Internet, you can monitor the number of Internet connection requests and the connection failure rate. Alerts are generated if the number of Internet connection requests and the connection failure rate exceed the specified thresholds. The alert notification must include the following information: the source IP address, the region of the source IP address, and whether the source IP address is malicious. In the following query statement, alerts are generated if the number of Internet connection requests exceeds 10 and the connection failure rate exceeds 50%.

- Sample query statement:

```
* | select ip as "Source IP address", total as "Number of Internet connection requests", round(rate * 100, 2) as "
Connection failure rate", failCount as "Number of unauthorized Internet connection requests", CASE when sec
urity_check_ip(ip) = 1 then 'yes' else 'no' end  as "Whether the IP address is malicious",  ip_to_country(ip) as "C
ountry", ip_to_province(ip) as "Province", ip_to_city(ip) as "City", ip_to_provider(ip) as "ISP" from (select CAS
E WHEN json_array_length(sourceIPs) = 1 then json_format(json_array_get(sourceIPs, 0)) ELSE  sourceIPs END
as ip, count(1) as total,
sum(CASE WHEN "responseStatus.code" < 400 then 0
ELSE 1 END) * 1.0 / count(1) as rate,
count_if("responseStatus.code" = 403) as failCount
from log  group by ip limit 10000) where ip_to_domain(ip) != 'intranet'  having "Number of Internet connection
requests" > 10 and "Connection failure rate × 100" > 50 ORDER by "Number of Internet connection requests" d
esc limit 100
```

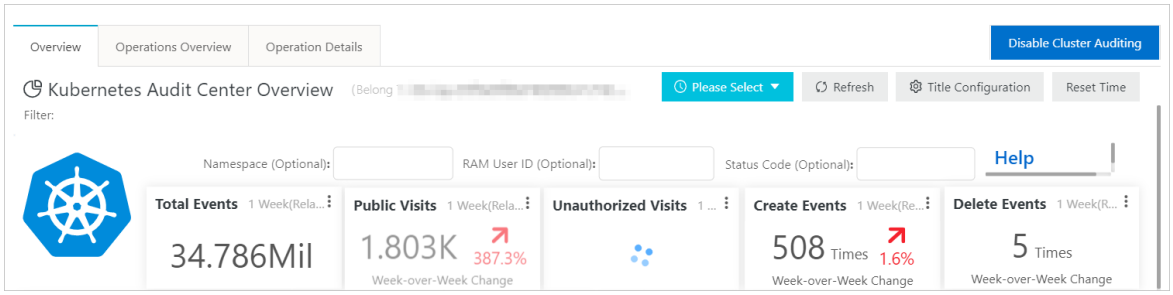
- The conditional expression is `Source IP address =~ ".*"` .

Enable cluster auditing

By default, **Enable Log Service** is selected when you create a cluster. In this case, kube-apiserver automatically collects audit log data for the cluster. If cluster auditing is disabled, perform the following steps to enable this feature:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Clusters > Clusters**. On the Clusters page, find the cluster for which you want to enable cluster auditing, and click **Manage**.
5. In the left-side navigation pane of the cluster details page, choose **Security > Cluster Auditing**. If cluster auditing is disabled, you are prompted to enable this feature.
6. Click **Enable Cluster Auditing Now**. **Select an existing project** or **create a project**, and then click **OK**.

   If the following page appears, cluster auditing is enabled.



Change the Log Service project

If you want to migrate the audit log data to another Log Service project, you can use the **Change Log Service Project** feature in cluster auditing.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the cluster details page, choose **Security > Cluster Auditing**.
5. In the upper-right corner of the Cluster Auditing page, click **Change Log Service Project**. Then, you can migrate the audit log data to another Log Service project.

Disable cluster auditing

You can perform the following steps to disable cluster auditing:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the cluster details page, choose **Security > Cluster Auditing**.
5. In the upper-right corner, click **Disable Cluster Auditing**.

Billing rules

- On the Overview tab of the Bills page, you can view the billing information about audit log data. For more information, see Bills.
- For more information about the billing methods of cluster auditing, see Pay-as-you-go.

Support for third-party logging services

You can find the source log file in the */var/log/kubernetes/kubernetes.audit* path of a master node. This file is in standard JSON format. When you create a cluster, you can specify a third-party logging service to collect and retrieve log data.

10.2.3. Enable service account token volume projection

When you create a cluster of Container Service for Kubernetes (ACK), you can enable **service account token volume projection** to enhance security when you use service accounts. This feature enables kubelet to request and store the token on behalf of the pod, and allows you to configure token properties such as the audience and validity period. As the token approaches expiration, kubelet proactively rotates the token if it is older than 80 percent of its time to live (TTL) or if it is older than 24 hours. This topic describes how to configure and use service account token volume projection for a cluster of Container Service for Kubernetes (ACK).

Context

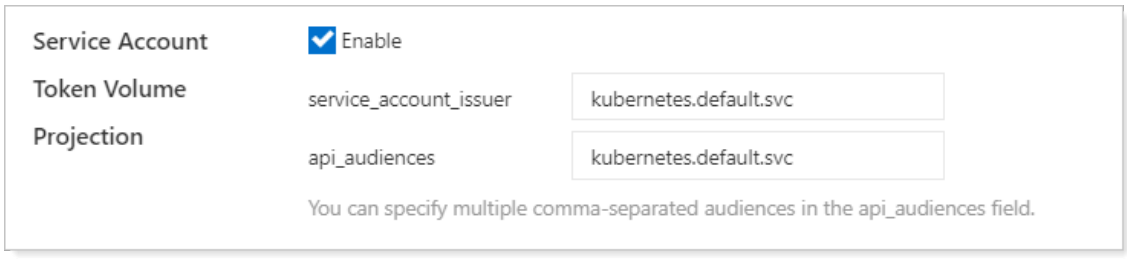
Service accounts are used to provide an identity for pods when they communicate with the API server. Traditionally, you may encounter the following security issues when you use service accounts:

- The JSON Web Tokens (JWTs) used by service accounts are not audience bound. A user of a service account can masquerade as another user and launch masquerade attacks.
- The service account token is stored in a Secret and delivered as a file to the corresponding node. A service account may be granted advanced permissions when a powerful system component is running. This results in a broad attack surface for the Kubernetes control plane. Attackers can obtain the service account used by this system component to launch privilege escalation attacks.
- JWTs are not time bound. A JWT that is compromised in the aforementioned attacks stays valid for as long as the service account exists. You can mitigate the issue only through service account signing key rotation, which is not supported by client-go and not automated by the control plane. Therefore, a complex operations process is required.
- A Kubernetes Secret must be created for each service account. This may put strains on elasticity and capacity in large-scale workload deployments.

ACK supports the BoundServiceAccountTokenVolume feature to enhance the security of service accounts. This feature enables pods to use a projected volume to mount the service account to a container, thus avoiding the dependency on Secrets.

Step 1: Enable service account token volume projection

When you create an ACK cluster, select **Enable** to enable **service account token volume projection**. For more information, see Create a professional managed Kubernetes cluster.



After service account token volume projection is enabled, the apiserver and controller-manager system components automatically enable the BoundServiceAccountTokenVolume feature gate and the following parameters are added to apiserver startup parameters.

Parameter	Description	Default value	Configuration in the ACK console
service-account-issuer	The issuer of the service account token, which corresponds to the iss field in the token payload.	kubernetes.default.svc	Supported.
api-audiences	The identifiers of the API, which are used to validate the tokens at the API server side.	kubernetes.default.svc	Supported. You can set one or more audiences. Separate multiple audiences with commas (,).
service-account-signing-key-file	The file path of the private key that signs the token.	/etc/kubernetes/pki/sa.key	Not supported. Default value: /etc/kubernetes/pki/sa.key.

Step 2: Use service account token volume projection

To configure a token with an audience of vault and a validity period of two hours for a pod, you can run the following command and use the following PodSpec template.

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
 name: build-robot
EOF
```

```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 containers:
 - image: nginx
  name: nginx
  volumeMounts:
  - mountPath: /var/run/secrets/tokens
    name: vault-token
 serviceAccountName: build-robot
 volumes:
 - name: vault-token
  projected:
   sources:
   - serviceAccountToken:
      path: vault-token
      expirationSeconds: 7200
      audience: vault
```

Notice

- Make sure that the pod periodically reloads the token when it rotates. This ensures that the pod can obtain the latest token in real time. We recommend that you reload the token once every five minutes. client-go 10.0.0 and later support automatic reloading to obtain the latest token.
- The permissions of the token file that corresponds to a service account are no longer 644. When BoundServiceAccountTokenVolume is used, the permissions are 600. When fsGroup is used, the permissions are 640.

Related information

- Service Account Token Volume Projection
- Configure Service Accounts for Pods

10.2.4. Use pod security policies

A pod security policy is an admission controller resource that validates requests to create and update pods in your cluster based on the rules that are defined by the policy. If a request to create or update a pod does not meet the rules, the request is rejected and an error is returned. This topic describes how to use pod security polices in a Container Service for Kubernetes (ACK) cluster.

Prerequisites

Before you configure network policies, make sure that you have performed the following steps:

- An ACK cluster is created.
- Connect to Kubernetes clusters by using kubectl.

The default pod security policy

By default, pod security policy control is enabled for standard managed Kubernetes clusters (Kubernetes 1.16.6) and standard dedicated Kubernetes clusters (Kubernetes 1.16.6). A pod security policy named ack.privileged is automatically created. This security policy accepts all types of pods. This provides the same effect as when pod security policy control is disabled for the cluster.

Query the default pod security policy | Query details about the default pod security policy

es the pod security policy, and the related ClusterRole and ClusterRoleBinding resources. ❯

Delete the ClusterRoleBinding resource that is related to the default pod security policy

Warning Before you delete the ClusterRoleBinding resource, you must configure a custom pod security policy and a related RBAC binding. Otherwise, all users, controllers, and service accounts will be unable to create or update pods.

After you configure a custom pod security policy and a related RBAC binding, you can delete the ClusterRoleBinding resource of the default pod security policy ack.privileged to enable the custom pod security policy.

Notice Do not delete or rename ack.privileged and the ack:podsecuritypolicy:privileged ClusterRole. These two resources are required to run the cluster.

tes the ClusterRoleBinding resource of the default pod security policy ack.privileged. ❯

Configure or restore the default pod security policy

code that configures or restores the default pod security policy and its RBAC binding. ❯

Related information

- Pod Security Policies

10.2.5. Use security-inspector to audit the CIS Kubernetes Benchmark

The Center for Internet Security (CIS) publishes the CIS Kubernetes Benchmark as a set of security recommendations for configuring Kubernetes in a secure manner. This topic describes how to use the security-inspector component to audit the CIS benchmark by using a command-line interface (CLI).

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- The security-inspector component is installed in the cluster. For more information, see Manage system components.

Overview of CIS Benchmarks

The Center for Internet Security develops CIS benchmarks, which are sets of best practices for the secure configuration of common systems. CIS Benchmarks are developed through a consensus-based process comprised of cybersecurity professionals and experts, and are widely accepted by governments, businesses, industries, and academia.

The CIS Kubernetes Benchmark is written for the open source Kubernetes distribution and intended to be as universally applicable across distributions as possible. The Benchmark versions are tied to specific Kubernetes versions. For more information, see CIS Kubernetes Benchmark.

CIS also releases CIS Kubernetes benchmarks that are specifically designed for Kubernetes distributions of different cloud service providers. For example, the CIS Benchmark for Alibaba Cloud Container Service for Kubernetes (ACK).

Use security-inspector to audit the CIS Kubernetes Benchmark

ACK allows you to use security-inspector to scan an ACK cluster based on the CIS Kubernetes Benchmark and obtain the scan report in CSV format. To do this, perform the following steps:

1. Run the following commands to create a scan task.

   To audit the CIS Kubernetes Benchmark, the component selects an appropriate benchmark version based on the Kubernetes version of the cluster.

   ```
   kubectl apply -f - <<EOF
   ---
   apiVersion: securityinspector.alibabacloud.com/v1alpha1
   kind: BenchmarkTask
   metadata:
    name: cis-kubernetes-benchmark
   spec:
    benchmarkVersion: 'cis-kubernetes-auto'
   ---
   apiVersion: securityinspector.alibabacloud.com/v1alpha1
   kind: BenchmarkJob
   metadata:
    name: cis-kubernetes-benchmark
   spec:
    taskName: cis-kubernetes-benchmark
   EOF
   ```

   You can set the benchmarkVersion parameter to one of the following values. Select the appropriate value based on your requirements. We recommend that you select cis-kubernetes-auto.

benchmarkVersion	Description	Applicable cluster
cis-kubernetes-auto	The component scans the cluster based on an appropriate CIS Kubernetes Benchmark that is automatically selected based on the Kubernetes version of the cluster.	Clusters of Kubernetes 1.15 and later
cis-kubernetes-ack-1.0	The component scans the cluster based on the CIS Benchmark for Alibaba Cloud Container Service for Kubernetes (ACK) 1.0.	Dedicated and managed ACK clusters
cis-kubernetes-1.6	The component scans the cluster based on the CIS Benchmarks for Kubernetes 1.6.	Clusters of Kubernetes 1.16 and later
cis-kubernetes-1.5	The component scans the cluster based on the CIS Benchmarks for Kubernetes 1.5.	Kubernetes 1.15

2. Wait for 5 minutes. Then, run the following command to check whether the scan task is completed.

   ```
   kubectl get benchmarkjobs.securityinspector.alibabacloud.com cis-kubernetes-benchmark -o 'jsonpath={.status.phase}'; echo
   ```

   If the output shows `Succeeded`, it indicates that the scan task is completed.

3. After the scan task is completed, run the following commands to obtain the scan report in CSV format.

```
for name in $(kubectl get benchmarkcsvresults.securityinspector.alibabacloud.com -l securityinspector.task
.name=cis-kubernetes-benchmark -o name)
do
 filename="cis-$(echo $name | awk -F '/' '{print $2}')"; \
 kubectl get $name -o jsonpath='{.result.data}' > "$filename".csv; \
 echo "saved $filename.csv"
done
```

When you read the scan report, you can determine whether remediation measures are required based on your business scenarios. For more information about the scan report, see Report interpretation.

Report interpretation

The following table describes the columns in the scan report.

Column name	Description	Whether measures are required
Date	The time of the scan.	No
Result Schema	The CIS benchmark based on which the scan is performed. Valid values: • cis-kubernetes-ack-1.0: the CIS Benchmark for Alibaba Cloud Container Service for Kubernetes (ACK) 1.0. • cis-kubernetes-1.6: the CIS Benchmarks for Kubernetes 1.6. • cis-kubernetes-1.5: the CIS Benchmarks for Kubernetes 1.5. For more information about the benchmarks, see CIS Kubernetes Benchmarks.	No
Node Name	The cluster node for which the report is generated.	No
Total Fail	The number of scored items that do not comply with benchmark recommendations.	For more information, see the description of **Result**.
Total Warn	The number of items that are not scored but require your attention.	For more information, see the description of **Result**.
Total Pass	The number of items that comply with benchmark recommendations.	No
Section Id	The section ID defined in the CIS benchmark.	No
Section Description	The section description defined in the CIS benchmark.	No
Test Id	The test ID defined in the CIS benchmark.	No
Test Description	The test description defined in the CIS benchmark.	No

Column name	Description	Whether measures are required
Scored	Whether the item is scored. Valid values: • Scored: Failure to comply with the recommendations decreases the final benchmark score. • Not scored: Failure to comply with the recommendations does not decrease the final benchmark score.	No
Test Remediation	The recommended remediation measure if the item does not comply with the benchmark recommendation. For more information, see CIS Kubernetes Benchmarks.	For more information, see the description of **Result**.
Result	The check result. Valid values: • fail: The scored item does not comply with the benchmark recommendation. • warn: The item is not scored but requires your attention. • pass: The item complies with the benchmark recommendation.	You can take the following measures based on the check result: • fail: We recommend that you take the remediation measures displayed in the **Test Remediation** column. You can also determine whether to adjust or fix the configurations based on actual business scenarios. • warn: You need to pay attention to the item and adjust the configuration based on actual business scenarios. • pass: No measure is required.

Related information

- CIS Kubernetes Benchmark
- security-inspector

10.2.6. Introduction to ack-kubernetes-webhook-injector

In some scenarios where fine-grained permission control is required, you may need to dynamically add the IP addresses of pods to security groups or Relational Database Service (RDS) whitelists. You may also need to remove these IP addresses from security groups or RDS whitelists. You can use ack-kubernetes-webhook-injector to perform these operations. This requires you to add annotations to pod configurations. This topic describes how to install and use ack-kubernetes-webhook-injector.

Prerequisites

- 创建Kubernetes托管版集群
- Connect to Kubernetes clusters by using kubectl

Context

In cloud computing scenarios, you must configure security policies to allow external access to some cloud resources. For example, you must add inbound rules to a security group to allow external access to the Elastic Compute Service (ECS) instances in the security group. You must configure an RDS whitelist to allow access from specified client IP addresses. When you create a Container Service for Kubernetes (ACK) cluster, you can add the CIDR block of the cluster nodes to an RDS whitelist. However, the following limits apply:

- The whitelist controls access in a coarse-grained manner because the IP addresses of all nodes and pods are added to the whitelist.
- The whitelist is not automatically deleted after the cluster is deleted. You must manually delete the whitelist.
- You cannot add inbound rules to the security group to which the cluster nodes belong when you create the cluster.

To fix the preceding issues, ack-kubernetes-webhook-injector is developed by ACK to provide fine-grained security control over cloud resources. You can use ack-kubernetes-webhook-injector to dynamically add the IP address of a pod to an RDS whitelist or security group. When the pod is deleted, the IP address is automatically removed from the RDS whitelist or security group.

The following features are provided by ack-kubernetes-webhook-injector:

- When a pod is created or deleted, the IP address of the pod is automatically added to or removed from a specified security group.
- When a pod is created or deleted, the IP address of the pod is automatically added to or removed from a specified RDS whitelist.

Install ack-kubernetes-webhook-injector

Before you use ack-kubernetes-webhook-injector, you must first install the component. Perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. Select and click **ack-kubernetes-webhook-injector**.
4. On the **Parameters** tab, set `use_aksk` to `true`, and specify the AccessKey pair of the current account or . For more information, see Obtain an AccessKey pair.

```
10 openapi:
11   # For test only. DO NOT use aksk authentication in production environments.
12   use_aksk: true
13   ak:
14   sk:
```

5. On the right side of the page, select a cluster to install the component and click **Create**.

Examples of using ack-kubernetes-webhook-injector

You only need to add an annotation to the Pod Spec parameter of the replication controller for a pod. The annotation must specify the ID of an RDS whitelist or security group. This way, when the pod is created, the IP address of the pod is automatically added to the specified RDS whitelist or security group. When the pod is deleted, the IP address of the pod is automatically removed from the specified RDS whitelist or security group.

The following annotations are supported:

- Annotations related to the RDS whitelist:
  - The ID of the RDS instance: ack.aliyun.com/rds_id.
  - The name of the RDS whitelist: ack.aliyun.com/white_list_name.
- The annotation related to the security group: ack.aliyun.com/security_group_id

In the following example, an RDS whitelist is used to show how to dynamically add the IP address of a pod to the RDS whitelist by using ack-kubernetes-webhook-injector.

1. Create a Deployment and add an annotation in the pod configuration to specify an RDS instance ID and add another annotation to specify an RDS whitelist. The following YAML template is provided as an example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
  app: inject-test
 name: inject-test
spec:
 replicas: 1
 selector:
  matchLabels:
   app: inject-test
 template:
  metadata:
   annotations:
    ack.aliyun.com/rds_id: <rm-wz9nanjcud75bxxxx>
    ack.aliyun.com/white_list_name: <rds_group>
   labels:
    app: inject-test
  spec:
   containers:
   - command:
    - sleep
    - "3600"
    image: alpine:latest
    name: inject-test
```

2. Run the following command to query the IP address of the application pod:

```
kubectl --kubeconfig .kube/config_sts_test -n inject-test get pod -o wide
```

The following output is returned:

```
NAME             READY     STATS    RESTARTS  AGE  IP        NODE
inject-test-68cc8f9bbf-gj86n  1/1      Running     0    22s  172.25.0.28  cn-hangzhou.xxx
```

The output shows that the IP address of the pod is 172.25.0.28.

3. Log on to the RDS console and check the whitelist of the specified RDS instance. For more information about how to check an RDS whitelist, see Configure an enhanced IP address whitelist.
4. Change the number of pod replicas to 0 for the Deployment that is created in Step 1. Then, log on to the RDS console and check the whitelist of the specified RDS instance.

   You can find that the IP address is removed from the RDS whitelist.

> **Note** You can perform similar steps to add the IP address of a pod to a security group by using ack-kubernetes-webhook-injector.

Uninstall ack-kubernetes-webhook-injector

If you no longer use ack-kubernetes-webhook-injector, you can uninstall ack-kubernetes-webhook-injector by using the release feature that is provided by ACK. For more information, see Delete a release. To delete the related configurations, we recommend that you run the following commands:

```
kubectl -n kube-system delete secret kubernetes-webhook-injector-certs
kubectl delete mutatingwebhookconfigurations.admissionregistration.k8s.io kubernetes-webhook-injector
```

10.2.7. Customize the SAN of the API server certificate for a managed Kubernetes cluster

The API server certificate of a Container Service for Kubernetes (ACK) cluster contains the Subject Alternative Name (SAN) field. By default, this field contains the domain name and IP address of the cluster. The elastic IP addresses (EIPs) and internal IP addresses of the Server Load Balancer (SLB) instances that are associated with the cluster are also included in this field. You can customize the SAN if you want to access the ACK cluster by using a proxy or through a different domain name. This topic describes how to add a SAN to the API server certificate when you create an ACK cluster. This topic also describes how to update the SAN of the API server certificate for an ACK cluster.

Prerequisites

A managed Kubernetes cluster is created. You can customize the SAN of the API server certificate for only managed Kubernetes clusters. Managed Kubernetes clusters include standard managed Kubernetes clusters and professional managed Kubernetes clusters. This section lists the methods that you can use to create three types of managed Kubernetes clusters. You need only to create one cluster in this example.

- 创建Kubernetes托管版集群
- Create a professional managed Kubernetes cluster
- Create a managed Kubernetes cluster with GPU-accelerated nodes

Context

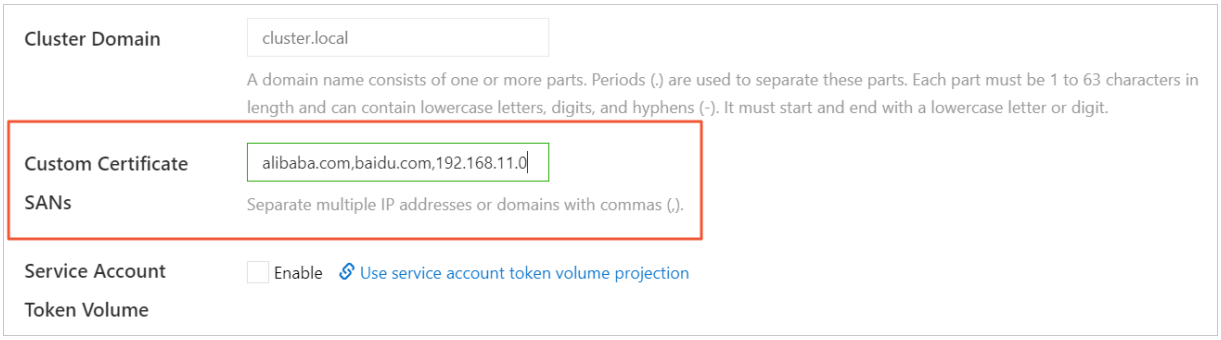
SAN is an extension of X.509. SAN allows you to associate various values with an SSL certificate by adding the values to the `subjectAltName` field. The values can be IP addresses, domain names, URIs, and email addresses.

Add a SAN to the API server certificate when you create an ACK cluster

This example describes how to add a SAN to the API server certificate when you create a managed Kubernetes cluster.

On the **Cluster Configurations** wizard page, click **Show Advanced Options**. In the **Custom Certificate SANs** field, enter the SAN that you want to add to the API server certificate. For more information, see 创建Kubernetes托管版集群.

> **Note** You can enter multiple values in the **Custom Certificate SANs** field. The values can be IP addresses, domain names, and URIs that comply with the conventions. Separate multiple values with commas (,).



In the preceding figure, two domain names and an IP address are entered in the **Custom Certificate SANs** field.

Update the SAN of the API server certificate for an ACK cluster

Notice After you update the SAN of the API server certificate for an ACK cluster, the API server is restarted. Perform this operation during off-peak hours.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the details page of the cluster, click the **Basic Information** tab and click **Update** on the right side of **Custom Certificate SANs**.
5. In the **Update Custom SAN** dialog box, set the **Custom Certificate SANs** parameter.
6. Click **OK**.

10.3. Security

10.3.1. Use a PSP

A pod security policy (PSP) is a Kubernetes resource that is used to validate pod creation requests against a set of rules. This feature can limit the container runtime behavior to enhance security. This topic describes how to create, manage, and associate a PSP to service accounts.

Prerequisites

- A managed or dedicated Kubernetes cluster is created. The cluster version is 1.14.8-aliyun.1 or later. For more information, see 创建Kubernetes托管版集群.
- An Alibaba Cloud account or a RAM user that has administrator permissions is used to log on to the Container Service for Kubernetes (ACK) console. For information about how to grant a RAM user administrator permissions, see Assign RBAC roles to a RAM user.

Context

ACK allows you to create custom PSPs by specifying parameters in policy templates. This feature can limit the container runtime behavior to enhance security. To ensure security, you may consider using security engines such as AppArmor and SELinux to validate pod creation requests. To meet your security requirements, ACK enables the PSP feature. The following list describes this feature:

- A PSP is a cluster-level Kubernetes resource model that is used to validate pod creation requests against a set of rules. To use a policy, you must install and enable the PSP admission controller for kube-apiserver. If a pod fails to meet the conditions defined in a specified policy, kube-apiserver rejects the pod creation

request.

- To use a policy, enable the PSP admission controller. For more information, see Enable an admission controller. You must also use RBAC to create a role that has the permission to use the policy and then bind the role to service accounts.
- By default, ACK enables the PSP feature for new managed or dedicated clusters. In this case, you do not have to configure related parameters for kube-apiserver. This reduces the O&M workloads.
- Assume that you have enabled the PSP feature for a cluster. To ensure the smooth running of the cluster, ACK creates a default policy named ack.privileged. This policy does not limit pod creation and is applied to all authenticated users of the cluster. However, we recommend that you create custom policies that grant users the minimum permissions. For example, you can create custom policies to prevent some users from creating privileged pods. You can also set the root file system to read-only or specify a list of host paths that can be used by hostPath volumes.

Create a custom policy

1. Log on to the ACK console.
2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. On the page that appears, choose **Security > PSPs**.
4. On the **PSPs** tab, click **Create**. In the dialog box that appears, select a policy template and click **Use**.

   The default policy ack.privileged does not limit pod creation and is applied to all authenticated users. You can use the following templates to create custom policies:

   - privileged-restricted: limits the creation of privileged pods. For example, you can specify whether to allow the creation of privileged pods that use the host network, host port, or host namespace.
   - privileged-root-volumes-restricted: limits the creation of privileged pods. For example, you can specify whether to allow the creation of privileged pods that use the host network, host port, or host namespace. You can also specify the supported volume types and limit the root privileges of pods.
5. On the **Create YAML** page, create a custom policy by specifying parameter values in the YAML file based on your needs. For more information about the parameters, see Pod security policies.

Parameter	Description
privileged	Specifies whether a privileged pod can be created.
hostPID and hostIPC	Specifies whether a pod can use the host namespace.
hostNetwork and hostPorts	Specifies whether a pod can use the host network and host ports.
volumes	The supported volume types.
allowedHostPaths	A list of host paths that can be used by hostPath volumes. After you specify pathPrefix, hostPath volumes can be stored in all paths that use the specified prefix.
allowedFlexVolumes	The FlexVolume drivers supported by the pod.
fsGroup	The ID of the FSGroup that owns the volumes mounted to the pod.
readOnlyRootFilesystem	Specifies whether the pod must run with a read-only root file system.
runAsUser, runAsGroup, and supplementalGroups	The IDs of the user, primary user group, and supplemental user group of the pod containers.

Parameter	Description
allowPrivilegeEscalation and defaultAllowPrivilegeEscalation	Specifies whether to allow privilege escalation for containers. The two parameters determine whether setuid binaries can be used.
defaultAddCapabilities, requiredDropCapabilities, and allowedCapabilities	The Linux capabilities of pod containers.
seLinux	The SELinux context of pod containers.
allowedProcMountTypes	The ProcMountTypes of pod containers.
annotations	The AppArmor or seccomp profile of pod containers.
forbiddenSysctls and allowedUnsafeSysctls	The sysctl profile of pod containers.

6. Click **OK**.
   On the **PSPs** tab, you can view the details of the policy that you create.

> **Note**
> - To modify a policy, click **YAML** in the **Actions** column.
> - To delete a policy, click **Delete** in the **Actions** column.

Associate a custom policy with service accounts

Assume that you have enabled the PSP feature for a cluster and disassociated the default policy. To use a custom policy, associate the policy with service accounts. If no custom policy is associated, the following error occurs when the system uses the service accounts to create a pod.

```
Error from server (Forbidden): error when creating xxx: pods "xxx" is forbidden: unable to validate against any pod security policy: []
```

You can associate a custom policy with specified or all service accounts in a specified namespace.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the page that appears, choose **Security > PSPs**.
5. Click the **Association Rules** tab, and click **Add Association Rule**.
6. In the **Add Association Rule** dialog box, specify **Namespace**, **Account**, and **PSP**.
7. Click **OK**.

Disassociate the default policy from service accounts

To enable service accounts in a namespace to use a custom policy, you must first disassociate the default policy from the service accounts.

- After you disassociate the default policy, all service accounts in the kube-system namespace and the system:nodes user group are authorized to use the default policy. This ensures the smooth running of system components and containers.

- All service accounts in a namespace are authorized to use the default policy if no custom policy is associated with the service accounts. This ensures that the containers in the namespace run smoothly.

> **Notice**
> - For a namespace in which service accounts are associated with a custom policy:
>   To ensure that the custom policy is effective, after you disassociate the default policy, the services accounts are not authorized to use the default policy. Therefore, before you disassociate the default policy, make sure that the service accounts in the namespace are associated with the desired custom policy.
> - For a namespace that is newly created:
>   The system does not automatically associate the default policy with service accounts in a new namespace. After you create a namespace, we recommend that you create a custom policy and associate it with service accounts in the namespace at the earliest opportunity.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the page that appears, choose **Security > PSPs**.
5. In the **PSP Management Guide** section, click **Disassociate Default PSP**.
6. In the **Disassociate** dialog box, click **OK**.

Enable the PSP admission controller

By default, ACK enables the PSP admission controller for new clusters. To ensure that PSPs take effect in an existing cluster, enable the admission controller for kube-apiserver.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the page that appears, choose **Security > PSPs**.
5. Enable the admission controller.
   - In the upper-right corner of the **PSPs** page, indicates that the admission controller is disabled.

     You can click **Change PSP State** and then click **OK** to enable it.
   - In the upper-right corner of the **PSPs** page,  indicates that the admission controller is enabled.

> **Notice** If you enable the admission controller, kube-apiserver is restarted. To minimize the impact on services, we recommend that you change the status of the admission controller during off-peak hours.

Examples

The following examples show how to create custom policies and associate them with service accounts.

1. Connect to Kubernetes clusters by using kubectl.
2. Create custom policies based on the privileged-restricted and privileged-root-volumes-restricted templates. For more information, see Create a custom policy.

3. Run the following commands to create two namespaces and service accounts in the namespaces, and use RBAC to bind required roles to service accounts.

```
kubectl create ns ack-all-restrictive
kubectl create ns ack-restrictive
kubectl create sa ack-dev -n ack-restrictive
kubectl create sa ack-tester -n ack-all-restrictive
kubectl -n ack-restrictive create rolebinding ack-dev-editor \
      --clusterrole=edit \
      --serviceaccount=ack-restrictive:ack-dev
kubectl -n ack-all-restrictive create rolebinding ack-tester-editor \
      --clusterrole=edit \
      --serviceaccount=ack-all-restrictive:ack-tester
```

4. On the **Association Rules** tab, associate custom policies with service account ack-dev.
   i. Click **Add Association Rule**. In the **Add Association Rule** dialog box, set **Namespace** to **ack-restrictive**, **Account** to **ack-dev**, and **PSP** to **privileged-restricted**. Click **OK**.
   ii. Click **Add Association Rule**. In the **Add Association Rule** dialog box, set **Namespace** to **ack-restrictive**, **Account** to **ack-dev**, and **PSP** to **privileged-root-volumes-restricted**. Click **OK**.
5. Run the following commands to generate aliases for service account ack-dev in namespace ack-restrictive and service account ack-tester in namespace ack-all-restrictive.

```
alias kubectl-dev='kubectl --as=system:serviceaccount:ack-restrictive:ack-dev -n ack-restrictive'
alias kubectl-tester='kubectl --as=system:serviceaccount:ack-all-restrictive:ack-tester -n ack-all-restrictive'
kubectl-dev auth can-i use  podsecuritypolicy/privileged-restricted
Warning: resource 'podsecuritypolicies' is not namespace scoped in group 'policy'
yes
```

6. Run the following command to create a privileged pod that uses the host network.

```
kubectl-dev apply -f- <<EOF
apiVersion: v1
kind: Pod
metadata:
 name: privileged
spec:
 hostNetwork: true
 containers:
  - name: busybox
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
EOF
Error from server (Forbidden): error when retrieving current configuration of:
Resource: "/v1, Resource=pods", GroupVersionKind: "/v1, Kind=Pod"
Name: "privileged", Namespace: "ack-restrictive"
Object: &{map["apiVersion":"v1" "kind":"Pod" "metadata":map["annotations":map["kubectl.kubernetes.io/
last-applied-configuration":""] "name":"privileged" "namespace":"ack-restrictive"] "spec":map["containers
":[map["command":["sh" "-c" "sleep 1h"] "image":"busybox" "name":"busybox"]] "hostNetwork":%! q(bool
=true)]]}
from server for: "STDIN": pods "privileged" is forbidden: User "system:serviceaccount:ack-restrictive:ack-de
v" cannot get resource "pods" in API group "" in the namespace "ack-restrictive"
```

   An error occurs when you create the privileged pod.
7. Run the following command to create an unprivileged pod.

```
kubectl-dev apply -f- <<EOF
apiVersion: v1
kind: Pod
metadata:
 name: non-privileged
spec:
 containers:
  - name: busybox
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
EOF
pod/non-privileged created
```

The command output indicates that the unprivileged pod is created.

8. Run the following commands to create a pod that has root privileges and a pod that does not have root privileges.

```
kubectl-tester apply -f- <<EOF
apiVersion: v1
kind: Pod
metadata:
 name: root-pod
spec:
 containers:
  - name: busybox
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
EOF
kubectl-tester apply -f- <<EOF
apiVersion: v1
kind: Pod
metadata:
 name: non-root-pod
spec:
 containers:
  - name: busybox
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    securityContext:
     runAsUser: 1000
     runAsGroup: 3000
EOF
```

9. Run the following command to query the statuses of pods.

   The pod that has root privileges cannot be created. The custom policy associated with service account ack-tester prevents the creation of pods that have root privileges.
   Command:

```
kubectl-tester get po
```

   The following information is displayed:

```
NAME       READY  STATUS                       RESTARTS  AGE
non-root-pod  1/1   Running                      0      115s
root-pod     0/1   CreateContainerConfigError  0      24s
```

10. Run the following command to query the error details.

    Command:

    ```
    kubectl describe pod root-pod
    ```

    The following information is displayed:

    ```
    Warning  Failed   28s (x5 over 2m1s)  kubelet, cn-shenzhen.192.168.1.52  Error: container has runAsNonRoot and image will run as root
    ```

10.3.2. Use the inspection feature to check for security risks in the workloads of an ACK cluster

This topic describes how to use the inspection feature to check for security risks of the workloads in a Container Service for Kubernetes (ACK) cluster. This topic also describes how to view inspection reports. This way, you can detect potential risks in workloads at the earliest opportunity.

Prerequisites

A managed Kubernetes cluster or a dedicated Kubernetes cluster is created, and the version of the cluster is 1.14.8 or later. For more information, see 创建Kubernetes托管版集群.

Grant permissions to a RAM user

If you log on as a Resource Access Management (RAM) user, you must authorize the RAM user to access the specified Log Service project. Otherwise, you fail to access the specified Log Service project. For more information, see Use custom policies to grant permissions to a RAM user.

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": "acs:log:*:*:project/<The name of the specified Log Service project>/*",
      "Effect": "Allow"
    }
  ]
}
```

Inspect workloads in an ACK cluster

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation page of the details page, choose **Security > Inspections**.
5. (Optional)If you have not installed the inspection component, click **Install** below **Install**. If you have installed the inspection component, skip this step.
6. In the upper-right corner of the **Inspections** page, click **Inspect**.
7. After the inspection is complete, click the Refresh icon to view the inspection report.

8. (Optional)In the upper-right corner of the **Inspections** page, click **Configure Periodic Inspection**. In the panel that appears, you can enable or disable periodic inspections and configure inspection items.

Inspection details

The **Inspections** page provides a table to show the inspection results of different workloads. The following features are provided to display the inspection results:

- Displays the values of **Number of Passed Items** and **Number of Failed Items** for each inspected workload.
- Displays the passed and failed inspection items, description of each inspection item, and suggestions for security reinforcement on the inspection details page.
- Allows you to configure workload whitelists for inspection.
- Allows you to select filter options from the **Namespace**, **Workload Type**, and **Passed or Failed** drop-down lists to narrow down inspection results.

Inspection reports

An inspection report provides the results of the latest inspection, including the following information:

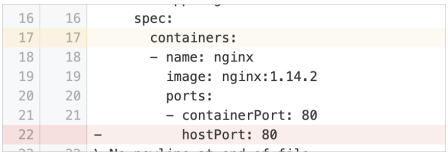
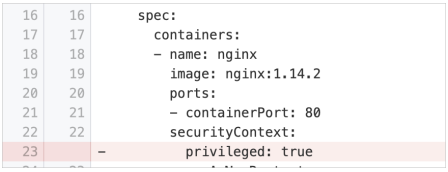
- Overview of the inspection results. This includes the total number of inspection items, the number and percentage of each inspected resource object, and the overall health status of the cluster.
- Statistics of the following inspection categories: health checks, images, networks, resources, and security.
- Detailed inspection results of each workload configuration. The results include resource categories, resource names, namespaces, inspection types, inspection items, and inspection results.

The following table describes the inspection items.

Inspection item	Inspection content and potential security risk	Suggestion

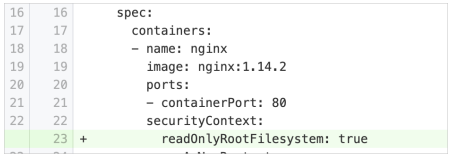
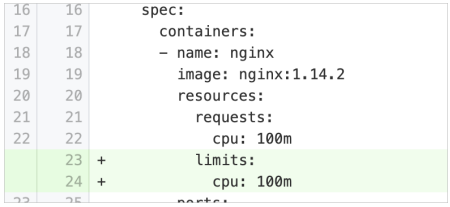
hostNetworkSet	Checks whether the pod specification of a workload contains the hostNetwork:true setting. This setting specifies that the pod uses the network namespace of the host. If the hostNetwork:true setting is used, the host network may be attacked by containers in the pod and the data transfer in the host network may be sniffed.	Delete the hostNetwork field from the pod specification.
		Example:
		`@@ -14,7 +14,6 @@ spec:`
		`labels:`
		`app: nginx`
		`spec:`
		`- hostNetwork: true`
		`containers:`
		`- name: nginx`
		`image: nginx:1.14.2`

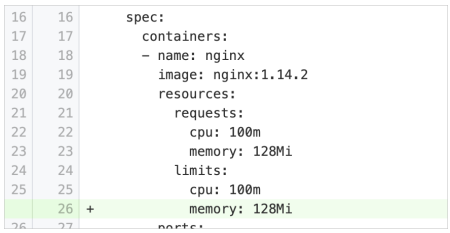
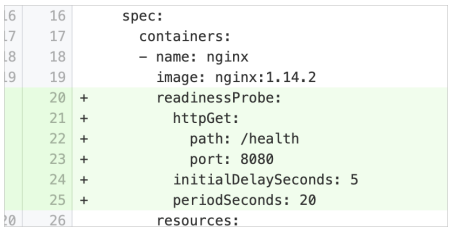
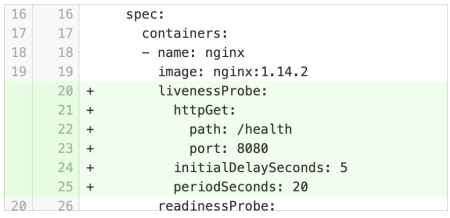
hostIPCSet	Checks whether the pod specification of a workload contains the hostIPC:true setting. This setting specifies that the pod uses the IPC namespace of the host. If the hostIPC:true setting is used, containers in the pod may attack the host processes and sniff process data.	Delete the hostIPC field from the pod specification.
		Example:
		`@@ -14,7 +14,6 @@ spec:`
		`labels:`
		`app: nginx`
		`spec:`
		`- hostIPC: true`
		`containers:`
		`- name: nginx`
		`image: nginx:1.14.2`

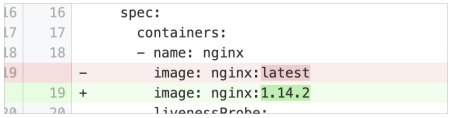
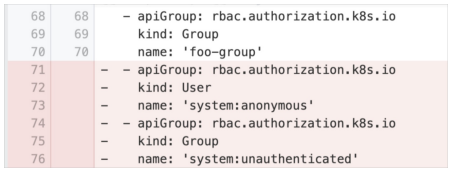
Inspection item	Inspection content and potential security risk	Suggestion
hostPIDSet	Checks whether the pod specification of a workload contains the hostPID:true setting. This setting specifies that the pod uses the PID namespace of the host. If the hostPID:true setting is used, containers in the pod may attack the host processes and collect process data.	Delete the hostPID field from the pod specification. Example: 
hostPortSet	Checks whether the pod specification of a workload contains the hostPort field. This field specifies the host port to which the listening port of the pod is mapped. If the hostPort field is specified, the specified host port may be occupied without authorization and the container port may receive unexpected requests.	Delete the hostPort field from the pod specification. Example: 
runAsRootAllowed	Checks whether the pod specification of a workload contains the runAsNonRoot:true setting. This setting specifies that containers in the pod are not allowed to run as the root user. If the runAsNonRoot:true setting is not used, malicious processes in the containers may intrude into your applications, hosts, or cluster.	Add the runAsNonRoot:true setting to the pod specification. Example: 
runAsPrivileged	Checks whether the pod specification of a workload contains the privileged:true setting. This setting specifies that containers in the pod are allowed to run in privileged mode. If the privileged:true setting is used, malicious processes in the containers may intrude into your applications, hosts, or cluster.	Delete the privileged field from the pod specification. Example: 

Inspection item	Inspection content and potential security risk	Suggestion
privilegeEscalationAllowed	Checks whether the pod specification of a workload contains the allowPrivilegeEscalation:false setting. This setting specifies that the child processes of a container are not granted higher privileges than the parent process. If the allowPrivilegeEscalation:false setting is not used, malicious processes in the container may be granted escalated privileges and perform unauthorized operations.	Add the allowPrivilegeEscalation:false setting to the pod specification. Example: `+ allowPrivilegeEscalation: false`

capabilitiesAdded	Checks whether the pod specification of a workload contains the capabilities field. This field is used to enable Linux capabilities for processes in containers. The capabilities include SYS_ADMIN, NET_ADMIN, and ALL. If the capabilities field is specified, malicious processes in the containers may intrude into your applications, cluster components, or cluster.	Modify the pod specification to retain only the required Linux capabilities and remove other capabilities.
		If processes in the containers do not require Linux capabilities, remove all Linux capabilities. Example:
		`- add:`
		`- - SYS_ADMIN`
		`- - NET_ADMIN`
		`- - KILL`
		`+ drop:`
		`+ - ALL`
		If processes in the containers require Linux capabilities, specify only the required Linux capabilities and remove other capabilities. Example:
		`- - SYS_ADMIN`
		`- - NET_ADMIN`
		`- - KILL`
		`+ - CHOWN`
		`+ drop:`
		`+ - ALL`

Inspection item	Inspection content and potential security risk	Suggestion
notReadOnlyRootFileSystem	Checks whether the pod specification of a workload contains the readOnlyRootFilesystem:true setting. This setting specifies that the root file system mounted to containers is read-only. If the readOnlyRootFilesystem:true setting is not used, malicious processes in the containers may modify the root file system.	Add the readOnlyRootFilesystem:true setting to the pod specification. If you want to modify files in a specific directory, set volumeMounts in the pod specification. Example:  If you want to modify files in a specific directory, set volumeMounts in the pod configuration. Example: 
cpuRequestsMissing	Checks whether the pod specification of a workload contains the resources.requests.cpu field. This field specifies the minimum CPU resources that are required to run each container. If the resources.requests.cpu field is not specified, the pod may be scheduled to a node that has insufficient CPU resources. This may lead to slow processes.	Add the resources.requests.cpu field to the pod specification. Example: 
cpuLimitsMissing	Checks whether the pod specification of a workload contains the resources.limits.cpu field. This field specifies the maximum CPU resources that can be used to run each container. If the resources.limits.cpu field is not specified, abnormal processes in containers may consume an excessive amount of CPU resources or exhaust the CPU resources of the node or cluster.	Add the resources.limits.cpu field to the pod specification. Example: 

Inspection item	Inspection content and potential security risk	Suggestion
memoryRequestsMissing	Checks whether the pod specification of a workload contains the resources.requests.memory field. This field specifies the minimum memory resources that are required to run each container. If the resources.requests.memory field is not specified, the pod may be scheduled to a node that has insufficient memory resources. As a result, processes in containers may be terminated when the Out of Memory (OOM) killer is triggered.	Add the resources.requests.memory field to the pod specification. Example: 
memoryLimitsMissing	Checks whether the pod specification of a workload contains the resources.limits.memory field. This field specifies the maximum memory resources that can be used to run each container. If the resources.limits.memory field is not specified, abnormal processes in containers may consume an excessive amount of memory resources or exhaust the memory resources of the node or cluster.	Add the resources.limits.memory field to the pod specification. Example: 
readinessProbeMissing	Checks whether the pod specification of a workload contains the readinessProbe field. This field specifies whether readiness probes are configured for containers. Readiness probes are used to check whether applications in the containers are ready to process requests. If the readinessProbe field is not specified, service exceptions may occur when requests are sent to applications that are not ready to process requests.	Add the readinessProbe field to the pod specification. Example: 
livenessProbeMissing	Checks whether the pod specification of a workload contains the livenessProbe field. This field specifies whether liveness probes are configured for containers. Liveness probes are used to check whether a container restart is required to resolve application anomalies. If the livenessProbe field is not specified, service exceptions may occur when application anomalies can be resolved only by restarting containers.	Add the livenessProbe field to the pod specification. Example: 

Inspection item	Inspection content and potential security risk	Suggestion
tagNotSpecified	Checks whether the image field in the pod specification of a workload specifies an image version or whether the value of the field is set to latest. If no image version is specified or the value of the field is set to latest, service exceptions may occur when containers use a wrong image version.	Modify the image field in the pod specification by specifying an image version. Set the field to a value other than latest. Example: 
anonymousUserRBACBinding	Checks role-based access control (RBAC) role bindings in the cluster and locates the configurations that grant access permissions to anonymous users. If anonymous users are allowed to access the cluster, they may gain access to sensitive information, attack the cluster, and intrude into the cluster.	Remove the configurations that grant access permissions to anonymous users from the RBAC role bindings. Example: 

Related information

- Configure a Security Context for a Pod or Container
- Configure Liveness, Readiness and Startup Probes

10.3.3. Use Runtime Security to monitor ACK clusters and configure alerts

Runtime Security monitors clusters of Alibaba Cloud Container Service for Kubernetes (ACK) and triggers alerts upon security events. Alerts are triggered upon the following security events: attacks by viruses or malware in containers and hosts, intrusions into containers, container escapes, and high-risk operations on containers. This topic describes how to use Runtime Security to monitor ACK clusters and configure alerts.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- Security Center is activated. For more information, see Purchase Security Center.
- If your account is a Resource Access Management (RAM) user, you must grant the RAM user the AliyunYundunSASReadOnlyAccess permission.

Context

Cloud-native applications are deployed in containers after they pass the authentication, authorization verification, and admission control of the API server. However, these applications are not protected from security risks. To resolve this issue, you can use Runtime Security. Runtime Security monitors applications and triggers alerts upon security events. Runtime Security is integrated with Security Center to detect vulnerabilities and raise alerts. This allows cluster administrators to monitor applications and receive alerts upon the following security events: attacks by viruses or malware in containers and hosts, intrusions into containers, container escapes, and high-risk operations on containers. You can view and manage alerts on the details page of an ACK cluster.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Security > Runtime Security**. On the **Runtime Security** page, you can view the monitoring information and alerts.
   - The security condition of the cluster appears on the **Runtime Security** page. The security condition includes security detection, node status, and defense capabilities.
     - Security detection: shows security events that have triggered alerts. Click **More** to navigate to the Security Center console for details.
     - Node status: shows healthy nodes and unhealthy nodes.
     - Defense Capabilities: shows pending alerts, the time when the anti-virus database was last updated, and the time when system vulnerabilities were last scanned.
   - On the **Runtime Security** page, click the **Alerts** tab to view the triggered alerts in real time. Alerts are triggered upon the following security events: attacks by viruses or malware in containers and hosts, intrusions into containers, container escapes, and high-risk operations on containers. For more information, see Overview.
     - On the Alerts tab, find the alert that you want to manage, and click **Manage** in the **Actions** column. In the dialog box that appears, add this alert to the whitelist or ignore this alert.
     - On the Alerts tab, find the alert that you want to view, and click **Details** in the **Actions** column. On the details page, you can view the information about this alert, such as the time when the event occurred, affected assets, and process IDs. On the **Details** page, click the **Diagnosis** tab. On the Diagnosis tab, you can enable automatic attack tracing and view the raw data.
   - On the **Runtime Security** page, click the **Vulnerabilities** tab to view vulnerabilities identified by Common Vulnerabilities and Exposures (CVE) on the cluster nodes.
     On the Vulnerabilities tab, find the vulnerability that you want to fix, and click **Fix** in the **Actions** column. On the page that appears, you can fix the vulnerability and repair the affected assets.

     > **Note** The vulnerabilities that are revealed on the Vulnerabilities tab include: Linux software vulnerabilities, Windows system vulnerabilities, web content management system (WCMS) vulnerabilities, application vulnerabilities, and emergency vulnerabilities. For more information, see the following topics:
     >
     > - View and handle Linux software vulnerabilities
     > - View and handle Windows system vulnerabilities
     > - View and handle Web-CMS vulnerabilities
     > - View and handle application vulnerabilities
     > - View and handle urgent vulnerabilities

10.4. FAQ about container security

This topic provides answers to some frequently asked questions about security groups.

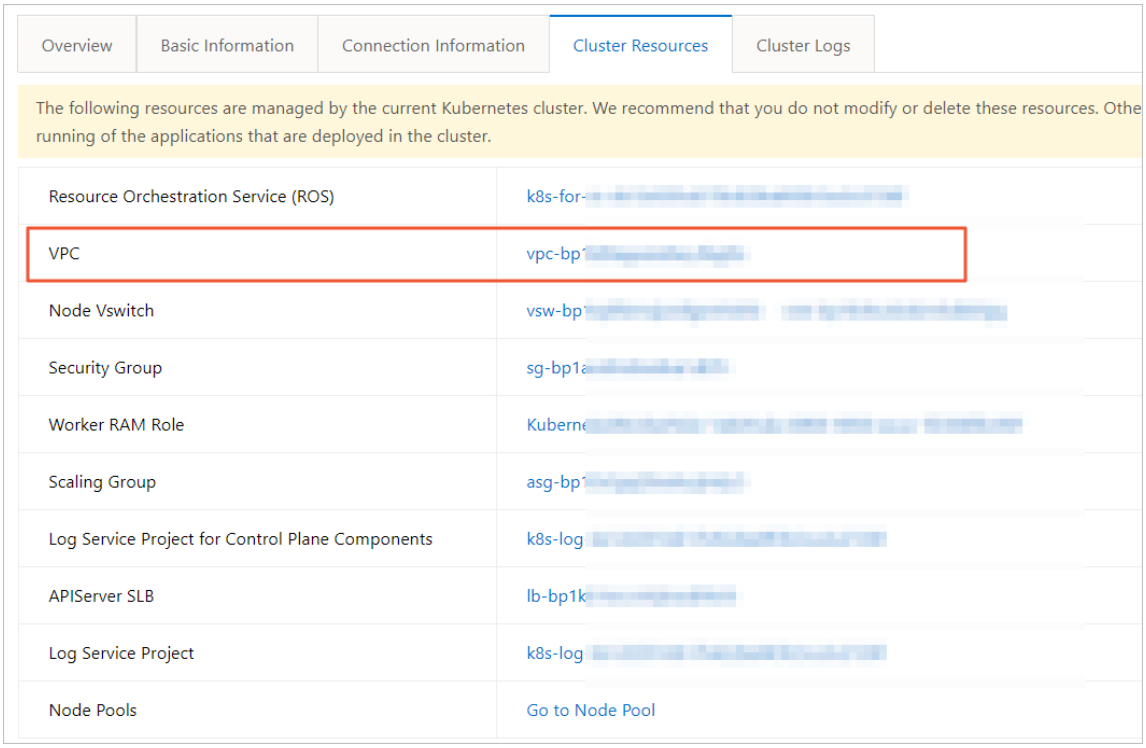
- Why do containers fail to communicate with each other?
- How do I specify a security group for an ACK cluster?

- Can I disable cluster auditing when I create a cluster or enable cluster auditing after the cluster is created?
- How do I renew the certificate of a dedicated Kubernetes cluster and renew the certificates of the components in the cluster?
- How do I fix the "no providers available to validate pod request" error during pod creation?
- Why am I unable to use existing Secrets in a new namespace?
- How do I fix the mount error when I mount the default token?
- How do I query the auditing log?
- How do I collect diagnostic data from nodes in an ACK cluster?
- How do I collect diagnostic data from nodes in an edge Kubernetes cluster?
- The kubelet log of an ACK cluster that uses the CentOS 7.6 operating system contains the "Reason:KubeletNotReady Message:PLEG is not healthy:" information

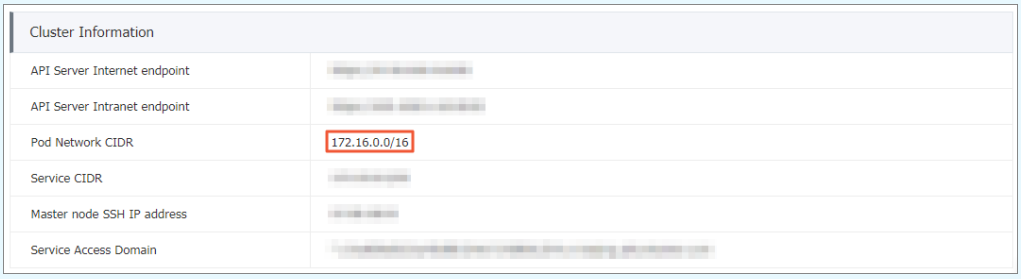
Why do containers fail to communicate with each other?

The following section describes the causes of network failures of different security group settings and provides solutions to the failures.

- Cause: The inbound rule in which **Authorization Object** is **Pod CIDR Block** and **Protocol Type** is **All** is deleted.
  i. Log on to the ACK console.
  ii. In the left-side navigation pane, click **Clusters**.
  iii. On the **Clusters** page, find the cluster that you want to manage and click the name or click **Details** in the **Actions** column.
  iv. On the **Cluster Resources** tab, click the link to the right of **VPC**.

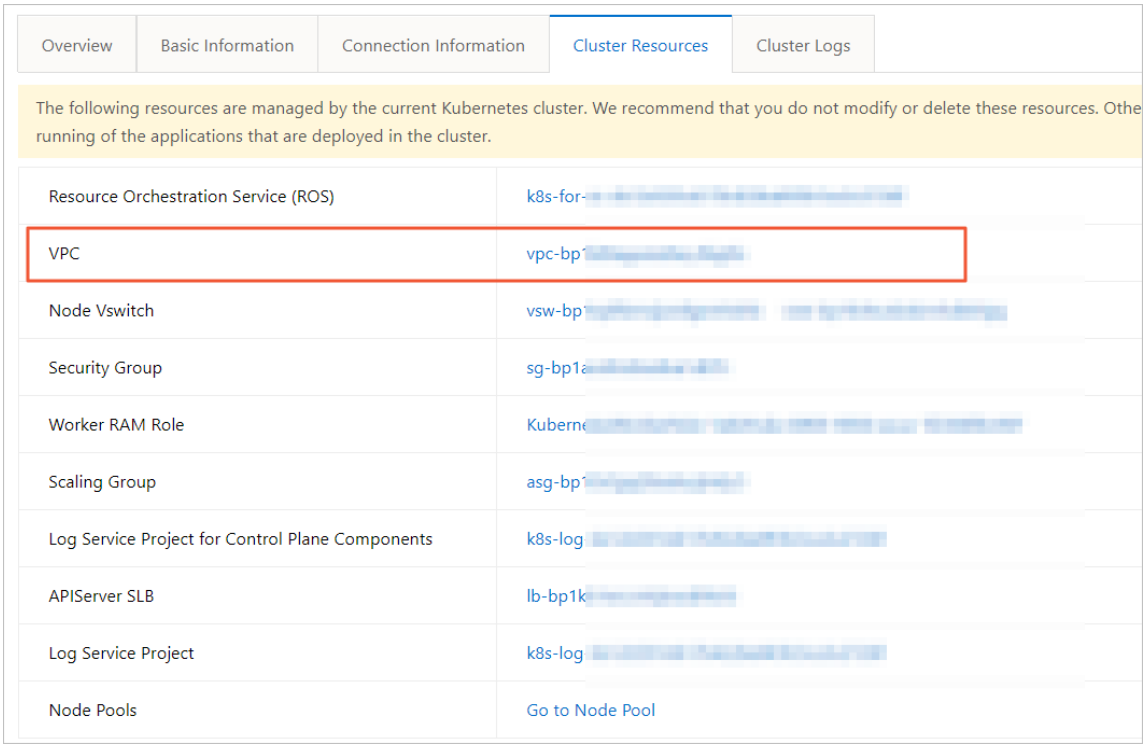


  v. On the **Resources** tab, click the number below **Security Group**.
  vi. Find the security group that you want to manage, and click **Add Rules** in the **Actions** column.
  vii. On the **Inbound** tab, click **Add Rule**.
  viii. Configure **Protocol Type**, **Port Range**, and **Authorization Object**. Then, click **Save**.

> **Note**
>
> - Set **Protocol Type** to **All**.
> - Set **Authorization Object** to the pod CIDR block of the cluster.
>   You can find the pod CIDR block in the **Cluster Information** section of the cluster details page in the ACK console.
>
> 
>
>
> For more information about **Authorization Object**, see Scenarios for security groupsConfiguration guide for ECS security groups.

The following figure shows the added inbound rule. The **Authorization Object** is the **Pod CIDR Block** of the cluster and **Protocol Type** is **All**.

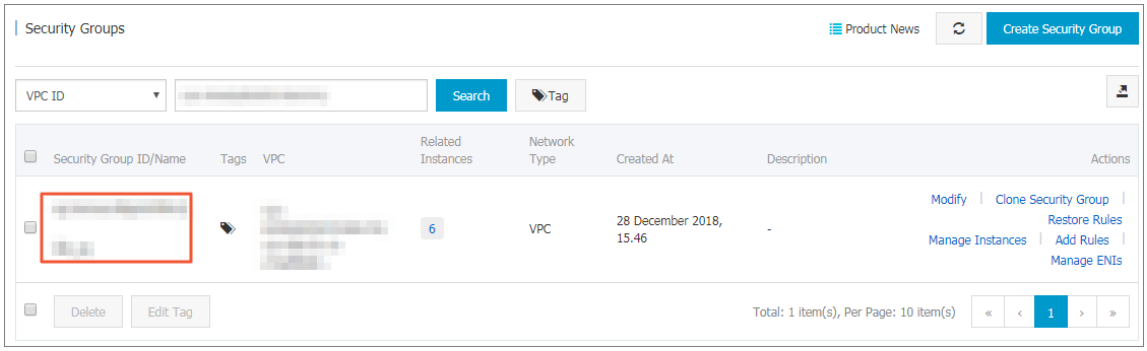
- Cause: The new Elastic Compute Service (ECS) instance and the Kubernetes cluster belong to different security groups.
  i. Log on to the ACK console.
  ii. In the left-side navigation pane, click **Clusters**.
  iii. On the **Clusters** page, find the cluster that you want to manage and click the name or click **Details** in the **Actions** column.
  iv. On the **Cluster Resources** tab, click the link to the right of **VPC**.



  v. On the **VPC Details** page, click the number below **Security Group** on the **Resources** tab. You are

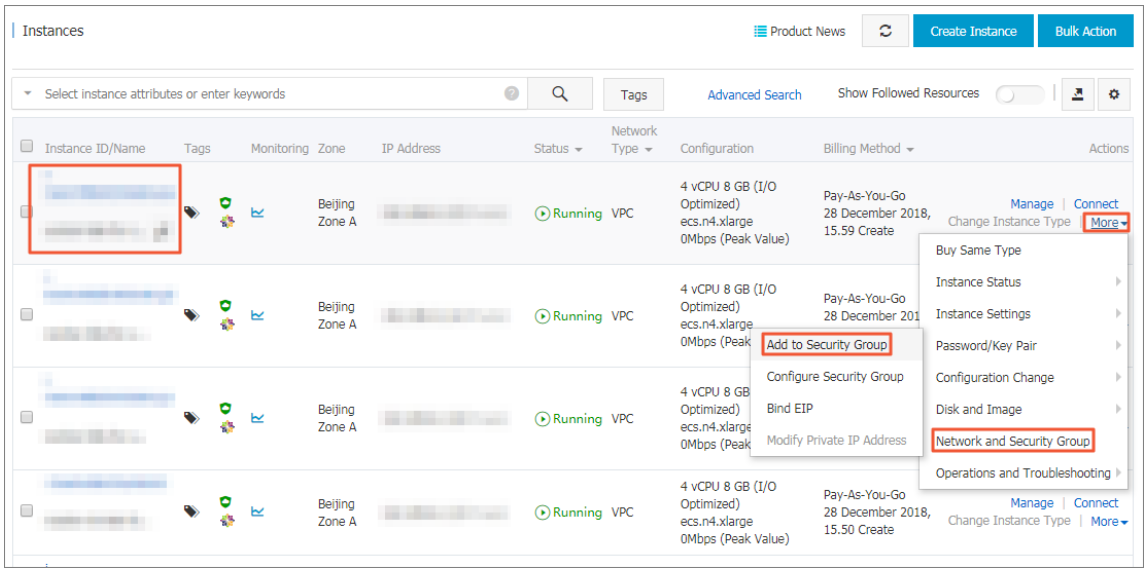
redirected to the **Security Groups** page in the ECS console. You can view the details of the security group on this page.

vi. On the **Security Groups** page, view the name of the security group.

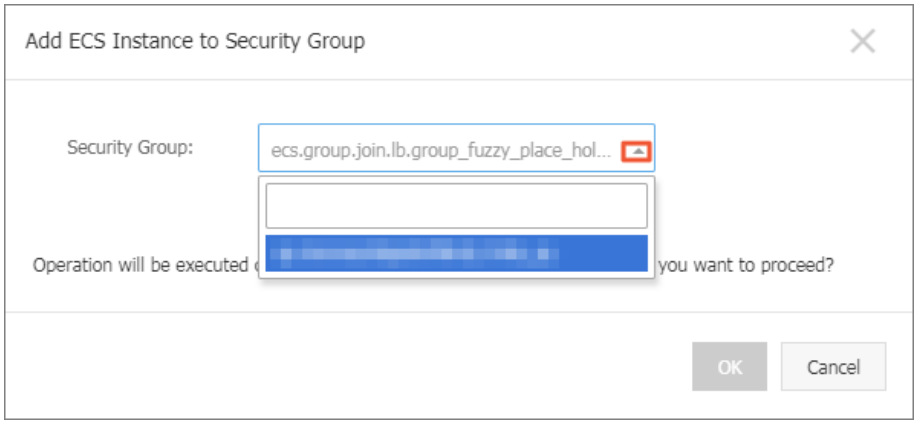


vii. In the left-side navigation pane of the ECS console, choose **Instances & Images > Instances**.

viii. On the **Instances** page, find the instance that you want to manage, and choose **More > Network and Security Group > Add to Security Group** in the **Actions** column. The **Add to Security Group** dialog box appears.



ix. In the **Security Group** drop-down list, enter the security group name that you obtained in Step 6.

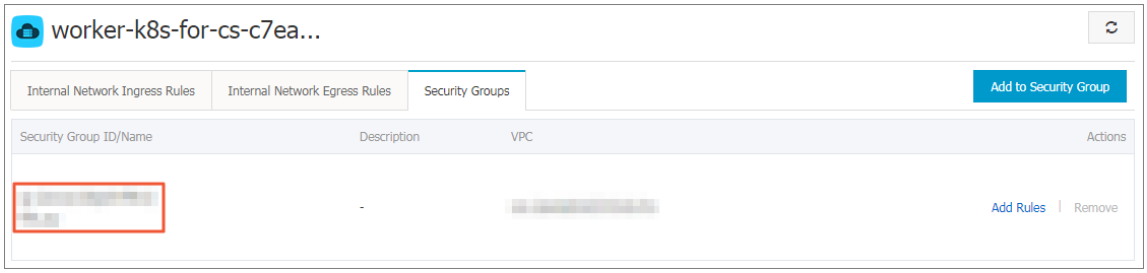


x. Click **OK**.

Verify the result

i. In the left-side navigation pane of the ECS console, choose **Instances & Images > Instances**. On the **Instances** page, click the name of the instance that is added to the security group.

ii. On the **Security Groups** tab, verify that the ECS instance is added to the security group to which the Kubernetes cluster belongs.



How do I specify a security group for an ACK cluster?

You cannot specify a security group for an ACK cluster. A default security group is automatically specified for an ACK cluster when the cluster is created. You can modify the rules of the default security group.

Can I disable cluster auditing when I create a cluster or enable cluster auditing after the cluster is created?

Yes. You can disable cluster auditing when you create a cluster and enable cluster auditing after the cluster is created. For more information, see Enable cluster auditing.

How do I renew the certificate of a dedicated Kubernetes cluster and renew the certificates of the components in the cluster?

- Approximately two months before a certificate expires, an internal message and SMS notification are sent to remind you about the expiration of the certificate. You can go to the clusters page in the console and click Renew to renew the certificate. For more information, see Update the Kubernetes cluster certificates that are about to expire.
- For more information about how to renew an expired certificate, see Update expired certificates of a Kubernetes cluster.

How do I fix the "no providers available to validate pod request" error during pod creation?

- If no custom pod security policy (PSP) is defined, the error appears because you deleted the default PSP. You can restore the default PSP to fix the error. For more information, see Use pod security policies.
- If you want to use a custom PSP, see Use a PSP.

Why am I unable to use existing Secrets in a new namespace?

Secrets are scoped to namespaces. You must create new Secrets in a new namespace.

How do I fix the mount error when I mount the default token?

The following error message is returned:

```
Normal Scheduled 13m default-scheduler Successfully assigned dev/alibaba-demo-67fcdbfb8-zklnp to cn-hangzhou.10.7.3.16  Warning FailedMount 13m (x2 over 13m) kubelet, cn-hangzhou.10.7.3.16 MountVolume.SetUp failed for volume 'default-token-8twx9' : mount failed: exit status 1 Mounting command: systemd-run Mounting arguments: --description=Kubernetes transient mount for /var/lib/kubelet/pods/62d39b35-9a4d-11ea-9870-c24d56a0e904/volumes/kubernetes.io~secret/default-token-8twx9 --scope -- mount -t tmpfs tmpfs /var/lib/kubelet/pods/62d39b35-9a4d-11ea-9870-c24d56a0e904/volumes/kubernetes.io~secret/default-token-8twx9 Output: Failed to start transient scope unit: Argument list too long  Warning FailedCreatePodContainer 3m40s (x49 over 13m) kubelet, cn-hangzhou.10.7.3.16 unable to ensure pod container exists: failed to create container for [kubepods burstable pod62d39b35-9a4d-11ea-9870-c24d56a0e904] : Argument list too long
```

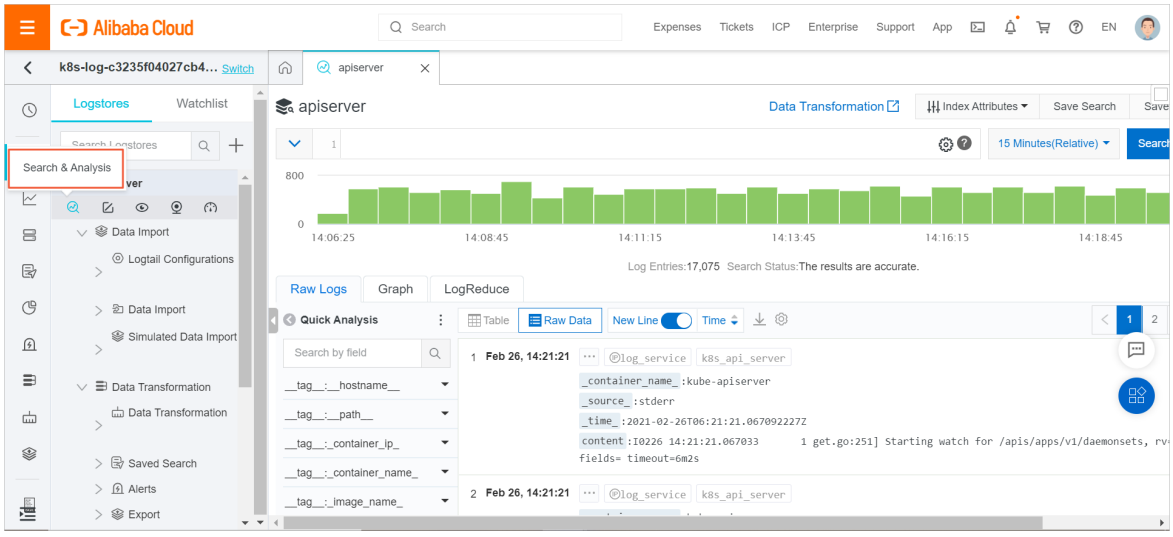
The systemd version is outdated.

- Upgrade systemd. For more information, see systemd.
- Run the `systemctl daemon-reload` command to reload units. For more information, see systemd.

How do I query the auditing log?

Query the auditing log about Role-Based Access Control (RBAC) operations

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the cluster information page, click the **Cluster Resources** tab. Then, click the link to the right of Log Service Project.
5. On the **Log Storage > Logstores** page, click the `audit-<cluster_id>` Logstore that you want to query and click **Search & Analysis**.



6. In the upper-right corner, click **15 Minutes(Relative)** to specify the time period that you want to query.

> **Note** Select a time period that covers the time when errors occurred. For example, 3 days, 7 day, or 15 days.

7. In the **Search & Analyze** search bar, enter the following SQL statement and then click **Search & Analyze**.

```
requestURI: "rbac.authorization.k8s.io" not (verb: get or verb: watch)
```

8. Click the  icon. In the **Log Download** dialog box, select **Download All Logs with Cloud Shell** and click **OK**.

Query the auditing log about ConfigMap operations

In the **Search & Analyze** search bar, enter the following SQL statement and click **Search & Analyze**. For more information, see How do I query the auditing log?.

```
requestURI: "configmaps" and <configmap_name> not (verb: get or verb: watch or verb: list)
```

> **Note** Replace *<configmap_name>* with the name of the ConfigMap that you want to query.

Query the auditing log about Deployment scaling operations

In the **Search & Analyze** search bar, enter the following SQL statement and click **Search & Analyze**. For more information, see How do I query the auditing log?.

```
requestURI: deployments and (verb: update or verb: patch) and replicas and deployments and <deployment_name> not deployment-controller
```

> **Note** Replace *<deployment_name>* with the name of the Deployment that you want to query.

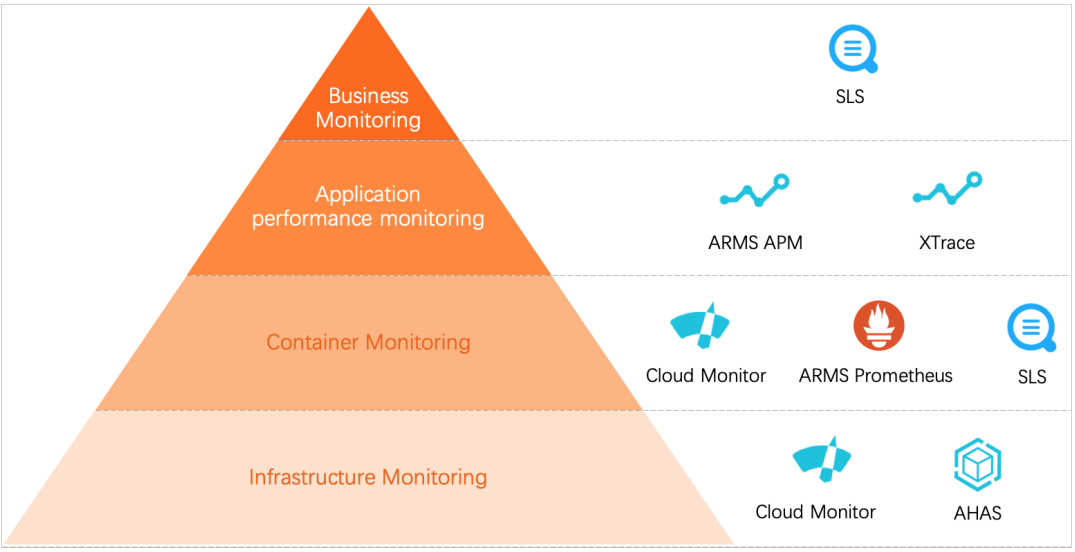
11.Observability

11.1. Overview of observability

Observability is the ability to infer internal states of a system based on external outputs of the system. The observability of Kubernetes includes monitoring and logging. Monitoring allows developers to keep track of the operation of a system and logging facilitates diagnostics and troubleshooting. This topic provides insights to the observability of Container Service of Kubernetes (ACK) and the observability of each layer. This helps you gain a comprehensive understanding of observability.

Observability of ACK

The observability of a system architecture that is built on top of ACK can be achieved at four layers. The four layers from bottom to top are: infrastructure, container performance, application performance, and business.



The following section describes the observability of each layer.

Infrastructure observability | Container performance observability | Application performance observability | Business observability

Related information

-
- Monitor basic resources
- Enable ARMS Prometheus
- Use Prometheus to monitor a Kubernetes cluster
- Event monitoring
- Monitor application performance
- Enable distributed tracing in ASM
- Get started with Tracing Analysis
- Collect log files from containers by using Log Service

11.2. Log management

11.2.1. Application log management

A Kubernetes cluster that runs on Alibaba Cloud Container Service provides you with multiple methods to manage application logs.

- Following the instructions of Collect log files from containers by using Log Service, you can make the best use of the functions provided by Alibaba Cloud Log Service, such as log statistics and analysis.
- With Log-pilot, an open source project provided by Alibaba Cloud Container Service, and 利用开源组件Log-pilot搭建Kubernetes日志解决方案, you can easily build your own application log clusters.

11.2.2. Collect log files from containers by using Log Service

Container Service for Kubernetes (ACK) is integrated with Log Service. When you create a cluster, you can enable Log Service to collect log files from containers, including standard outputs (stdout) and text files.

Step 1: Install Logtail

When you create a cluster, select **Enable Log service** to install Logtail. You can also install Logtail for an existing cluster.

Install Logtail when you create a cluster:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Create Kubernetes Cluster**.

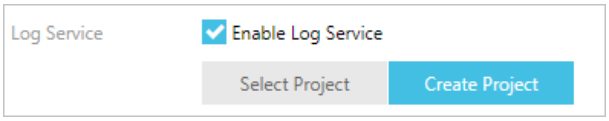
   In this example, only the steps to enable Log Service are described. For more information about how to create an ACK cluster, see 创建Kubernetes托管版集群.
4. In the **Component Configurations** step, select **Enable Log Service** to install Logtail for the Kubernetes cluster to be created.

   When the **Enable Log Service** check box is selected, the system prompts you to create a Log Service project. For more information about how log files are managed in Log Service, see Project). Select one of the following methods to specify a project in one of the following ways:

   - You can select an existing project to manage the collected log files.

   - You can click Create Project. Then, a project named `k8s-log-{ClusterID}` is automatically created to manage the collected log files. ClusterID indicates the unique ID of the cluster to be created.

5. After you set the parameters, click **Create Cluster** in the lower-right corner. In the message that appears, click **OK**.

   After the ACK cluster is created, you can find the cluster that has Logtail enabled on the Clusters page.

Install Logtail for an existing cluster:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**. In the **Logs and Monitoring** section, find **logtail-ds**.
5. Click **Install** next to **logtail-ds**.
6. In the **Note** message, click **OK**.

If an earlier version of Logtail is installed, click **Upgrade** next to **logtail-ds**.

Step 2: Configure Log Service when you create an application

You can configure Log Service to collect log files from containers when you create an application. You can use the wizard in the console or YAML templates to create applications.

Use the wizard in the console to create an application and configure Log Service

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, select a namespace from the **Namespace** drop-down list. Then, click **Create from Image** in the upper-right corner of the page.
6. In the **Basic Information** step, specify **Name**, **Replicas**, and **Type**. Then, click **Next** to go to the **Container** step.

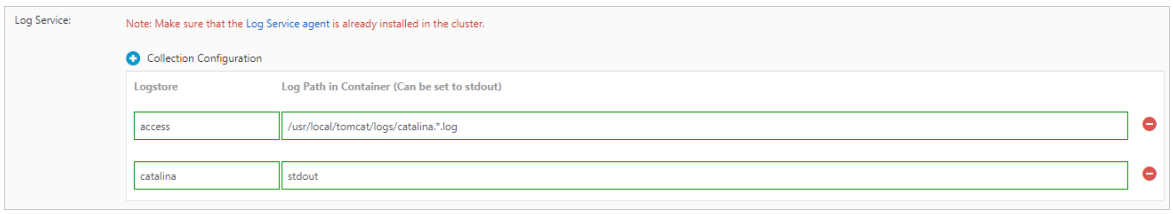
   The following steps describe how to configure Log Service. For more information about other configuration items, see Create a stateless application by using a Deployment.
7. In the **Log** section, click the plus sign (+) to add a Logstore. You must specify **Logstore** and **Log Path in Container (Can be set to stdout)**.
   - Logstore: specifies the Logstore that stores collected log files. If the specified Logstore does not exist, the system automatically creates a Logstore in the Log Service project that is associated with the cluster.

     > **Note** The name of the Logstore cannot contain underscores (_). You can use hyphens (-) instead.

   - Log Path in Container (Can be set to stdout): specifies the path from which you want to collect log files. For example, */usr/local/tomcat/logs/catalina.*.log* indicates that log files from the Tomcat application are collected.

     > **Note** If you set the value to stdout, standard outputs and error messages are collected.

     After you add a configuration entry, the system automatically creates a Logstore. By default, logtail-ds collects log files in simple mode. In this mode, log files are collected by line. To use more log collection methods, go to the Log Service console and modify the log collection configurations of the project and Logstore. By default, the project uses k8s-log as the prefix.



8. Add **Custom Tag**.

Click the plus sign (+) to add custom tags. Each tag is a key-value pair that is appended to the collected log files. You can use custom tags to mark log data. For example, you can use a tag to denote the application version.

9. After you set the other parameters, click **Next** to set advanced settings. For more information about advanced settings, see Create a stateless application by using a Deployment.

Create an application by using a YAML template

1. In the left-side navigation pane of the cluster details page, click **Workloads**.
2. On the **Deployments** page, select a namespace from the **Namespace** drop-down list. Then, click **Create from YAML** in the upper-right corner of the page.
3. The syntax of the YAML template is the same as that of Kubernetes. You can use `env` to define **collection configurations** and **custom tags**. You must also configure volumeMounts and volumes. The following is an example of a pod for collecting log files:

```
apiVersion: v1
kind: Pod
metadata:
 name: my-demo
spec:
 containers:
 - name: my-demo-app
  image: 'registry.cn-hangzhou.aliyuncs.com/log-service/docker-log-test:latest'
  env:
  ######### Configure environment variables ###########
  - name: aliyun_logs_log-stdout
   value: stdout
  - name: aliyun_logs_log-varlog
   value: /var/log/*.log
  - name: aliyun_logs_mytag1_tags
   value: tag1=v1
  ################################
  ######### Configure volume mounting ###########
  volumeMounts:
  - name: volumn-sls-mydemo
   mountPath: /var/log
 volumes:
 - name: volumn-sls-mydemo
  emptyDir: {}
 ################################
```

Perform the following steps in sequence based on your needs:

> **Note** If you have more requirements for log collection, see Step 3: Configure advanced parameters in the env field.

i. Add **collection configurations** and **custom tags** by using environment variables. All environment variables related to log collection must use `aliyun_logs_` as the prefix.

- Add log collection configurations in the following format:

```
- name: aliyun_logs_{Logstore name}
  value: {Log file path}
```

In the preceding YAML template, two environment variables are added to the log collection configuration. Environment variable `aliyun_logs_log-stdout` indicates that a Logstore named log-stdout is created to store the stdout files collected from containers.

> **Note** The name of the Logstore cannot contain underscores (_). You can use hyphens (-) instead.

- **Custom tags** must be specified in the following format:

```
- name: aliyun_logs_{Tag name without underscores (_)}_tags
  value: {Tag name }={Tag value}
```

After a tag is added, the tag is automatically appended to the collected log files when logtail-ds collects the container logs.

ii. If you specify a log path to collect log files that are not standard outputs, you must configure volumeMounts.
In the preceding YAML template, the mountPath field in volumeMounts is set to */var/log*. This allows logtail-ds to collect log files from the */var/log/*.log* directory.

4. After you modify the YAML template, click **Create** to submit the configurations.

(Optional)

Step 3: Configure advanced parameters in the env field

You can specify multiple parameters in the env field to configure log collection. The following table describes the parameters.

> **Notice** This configuration method is not applicable to edge computing scenarios.

Field	Description	Example	Precaution

Field	Description	Example	Precaution
aliyun_logs_{key}	• Required.{key} can contain only lowercase letters, digits, and hyphens (-). • If the specified aliyun_logs_{key}_logstore does not exist, a Logstore named {key} is created. • To collect standard outputs of container log files, set the value to stdout. You can also set the value to a path inside a container.	- name: aliyun_logs_catalina stdout - name: aliyun_logs_access-log /var/log/nginx/access.log	• By default, logtail-ds collects log files in simple mode. In this case, the collected log files are not parsed. If you want to parse the log files, we recommend that you change the collection mode in the Log Service console. For more information, see Use the console to collect Kubernetes text files in DaemonSet mode, Use the console to collect Kubernetes stdout and stderr outputs in DaemonSet mode, or Use CRDs to collect Kubernetes container logs in DaemonSet mode. • The value of {key} must be unique in the cluster.
aliyun_logs_{key}_tags	Optional. This parameter is used to add tags to log data. The value must be in the following format: {tag-key}={tag-value}.	- name: aliyun_logs_catalina_tags app=catalina	-
aliyun_logs_{key}_project	Optional. This parameter specifies a project in Log Service. By default, the project that you specified when you create the cluster is used.	- name: aliyun_logs_catalina_project my-k8s-project	The region of the project must be the same as where your logtail-ds is deployed.
aliyun_logs_{key}_logstore	Optional. This variable specifies a Logstore in Log Service. By default, the Logstore is named after {key}.	- name: aliyun_logs_catalina_logstore my-logstore	-

Field	Description	Example	Precaution
aliyun_logs_{key}_shard	Optional. This variable specifies the number of shards in the Logstore. Valid values: 1 to 10. Default value: 2.	- name: aliyun_logs_catalina_shard 4	-
aliyun_logs_{key}_ttl	Optional. This parameter specifies the number of days for which log data is retained. Valid values: 1 to 3650. • To permanently retain log data, set the value to 3650. • Default value: 90.	- name: aliyun_logs_catalina_ttl 3650	-
aliyun_logs_{key}_machinegroup	Optional. This parameter specifies the machine group of the application. By default, the machine group is the one where your logtail-ds is deployed.	- name: aliyun_logs_catalina_machinegroup my-machine-group	-

- **Scenario 1: Collect log files from multiple applications and store them in the same Logstore**
  In this scenario, set the aliyun_logs_{key}_logstore parameter. The following example shows how to collect standard outputs from two applications and store the outputs in stdout-logstore.
  Configure the following environment variables for Application 1:

```
######### Configure environment variables ###########
  - name: aliyun_logs_app1-stdout
    value: stdout
  - name: aliyun_logs_app1-stdout_logstore
    value: stdout-logstore
```

  Configure the following environment variables for Application 2:

```
######### Configure environment variables ###########
  - name: aliyun_logs_app2-stdout
    value: stdout
  - name: aliyun_logs_app2-stdout_logstore
    value: stdout-logstore
```

- **Scenario 2: Collect log files from different applications and store them in different projects**
  In this scenario, perform the following steps:
  i. Create a machine group in each project and set the machine group ID in the following format: k8s-group-{cluster-id}, where {cluster-id} is the ID of the cluster. You can customize the machine group name.
  ii. Specify the project, Logstore, and machine group in the environment variables for each application.

```
######### Configure environment variables ###########
  - name: aliyun_logs_app1-stdout
    value: stdout
  - name: aliyun_logs_app1-stdout_project
    value: app1-project
  - name: aliyun_logs_app1-stdout_logstore
    value: app1-logstore
  - name: aliyun_logs_app1-stdout_machinegroup
    value: app1-machine-group
```

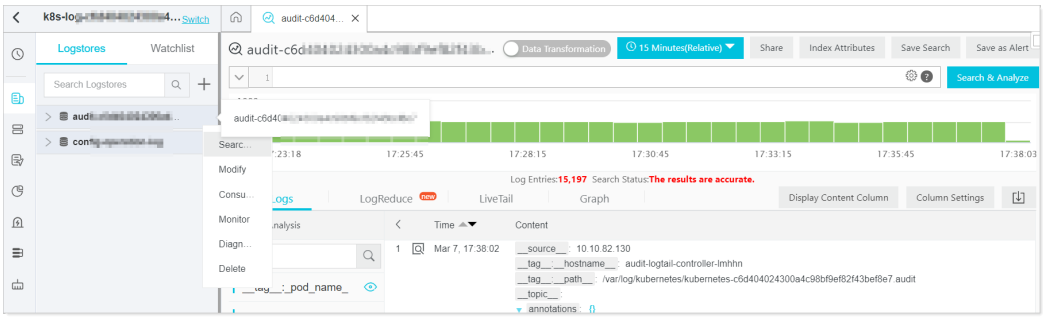
Step 4: View log files

The following example shows how to view log files of the Tomcat application created by using the wizard in the console. Log files from the Tomcat application are stored in Log Service. You can log on to the Log Service console to view log files collected from the containers. Perform the following steps to view log files:

1. Log on to the Log Service console.
2. In the **Projects** section, click the project that is associated with the Kubernetes cluster to go to the **Logstores** tab. By default, the project name is in the format of k8s-log-{Kubernetes cluster ID}.
3. In the Logstores list, find the Logstore that is specified when you configure log collection. Move the pointer over the Logstore name and click the

   icon. Then, click **Search & Analysis**.
4. In this example, you can view standard outputs of the Tomcat application and text log files of containers. You can also find that custom tags are appended to the collected log files.



After Log Service is enabled for the application, you can view log files of containers in the ACK console. Perform the following steps to view log files of containers:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Event Center**.
5. On the **Log Center** page, click the **Application Logs** tab and specify the filter conditions. Then, click **Select Logstore** to view log files of containers.

More information

1. By default, Log Service collects log files by line and does not parse the log files. If you want to change the log collection mode and parse the log files, modify the log collection configurations in the Log Service console. For more information, see the following topics:

- Use the console to collect Kubernetes text log files in DaemonSet mode
- Use the console to collect Kubernetes stdout and stderr outputs in DaemonSet mode

2. You can also use CustomResourceDefinitions (CRDs) to collect log files from Kubernetes clusters
3. For more information about troubleshooting, see Troubleshoot collection errors.

11.2.3. Configure Log4jAppender for Kubernetes and Log Service

This topic describes how to configure a YAML file to export the logs of a Container Service for Kubernetes (ACK) cluster to Log Service, without the need to modify the application. In addition, an application that can be managed through API operations is deployed in the ACK cluster. The application is used to generate logs for testing purposes.

Prerequisites

- ACK is activated and an ACK cluster is created.
  In this example, an ACK cluster is created in the China (Hangzhou) region.
- An AccessKey pair is created or Resource Access Management (RAM) is activated. Make sure that the required permissions are granted. In this example, an AccessKey pair is created.

Context

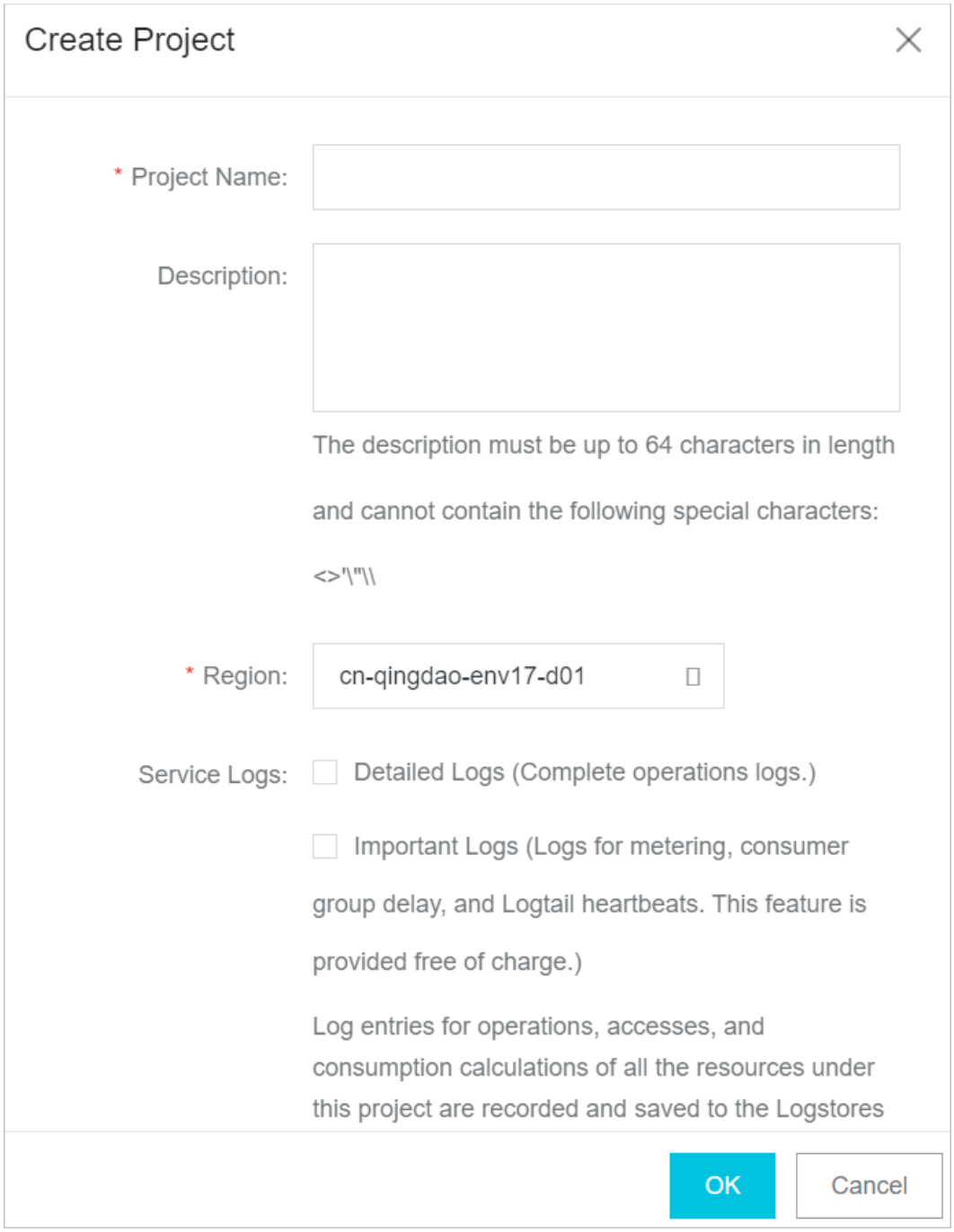
Log4j is an open source project of Apache. Log4j consists of three components: log level, log output destination, and log output format. You can configure Log4Appender to export log data to the console, a log file, a GUI component, a socket server, an NT event viewer, or a UNIX syslog daemon.

Procedure

1. Configure Log4jAppender in Log Service.
   i. Create a project in Log Service.

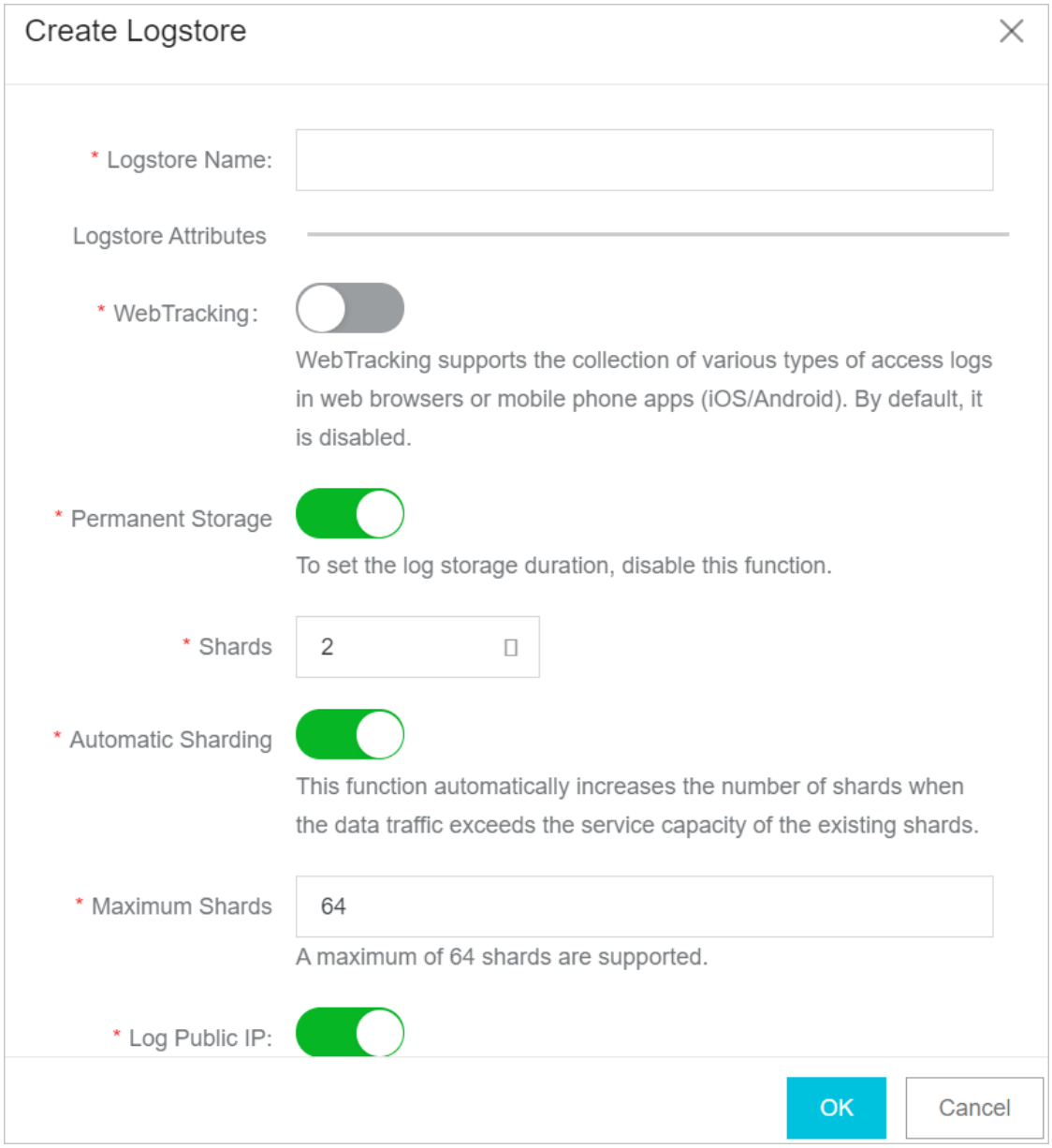
   In this example, a project named k8s-log4j is created in the China (Hangzhou) region where the ACK cluster is deployed. For more information, see Create a project.

   > **Note** We recommend that you create a project in the region where the ACK cluster is deployed. When a Log Service project and an ACK cluster are deployed in the same region, logs are transmitted within the internal network. This enables real-time collection and quick retrieval of log data. This also avoids cross-region transmission, which requires additional bandwidth and time costs.

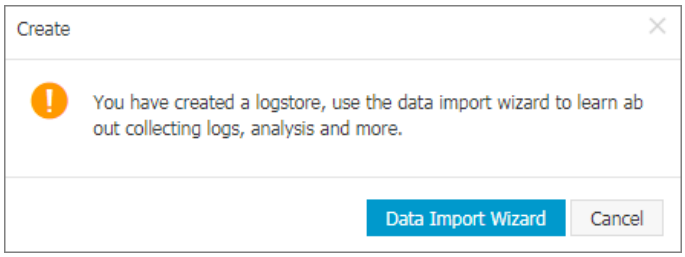

Create Project
* Project Name:
Description:
The description must be up to 64 characters in length and cannot contain the following special characters: <>'\"\\
* Region:
cn-qingdao-env17-d01
Service Logs:
Detailed Logs (Complete operations logs.)
Important Logs (Logs for metering, consumer group delay, and Logtail heartbeats. This feature is provided free of charge.)
Log entries for operations, accesses, and consumption calculations of all the resources under this project are recorded and saved to the Logstores
OK
Cancel

ii. Create a Logstore for the k8s-log4j project.

In this example, a Logstore named k8s-logstore is created. For more information, see Create a Logstore.

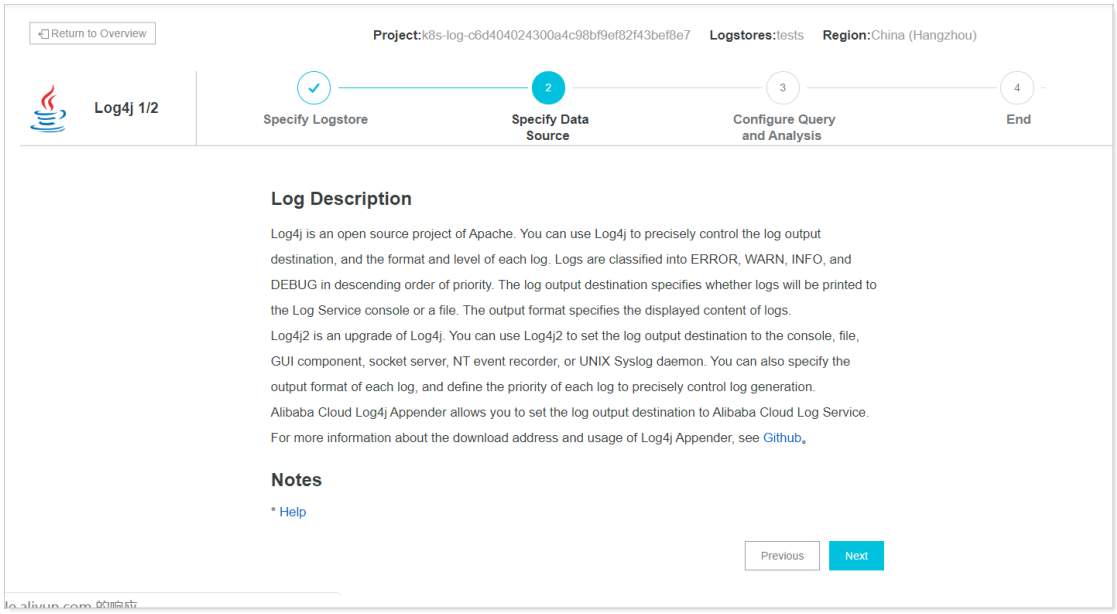


iii. After the k8s-logstore Logstore is created, a dialog box appears, which shows instructions on how to use the Data Import wizard.



iv. Select Log4jAppender under **Custom Code** and configure the settings by performing the steps that are provided on the page.

In this example, Log4jAppender is configured with the default settings. You can also customize the settings to meet your business requirements.



2. Configure Log4jAppender for the ACK cluster.

In this example, the demo-deployment and demo-Service files are used.

i. Connect to your ACK cluster.

For more information, see Use SSH to connect to a Kubernetes cluster or Connect to Kubernetes clusters by using kubectl.

ii. Obtain the *demo-deployment.yaml* file and configure the `JAVA_OPTS` environment variable.

The following is a sample template of the *demo-deployment.yaml* file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: log4j-appender-demo-spring-boot
 labels:
  app: log4j-appender
spec:
 replicas: 1
 selector:
  matchLabels:
   app: log4j-appender
 template:
  metadata:
   labels:
    app: log4j-appender
  spec:
   containers:
   - name: log4j-appender-demo-spring-boot
    image: registry.cn-hangzhou.aliyuncs.com/jaegertracing/log4j-appender-demo-spring-boot:0.0.2
    env:
     - name: JAVA_OPTS          ## Take note of the following fields
       value: "-Dproject={your_project} -Dlogstore={your_logstore} -Dendpoint={your_endpoint} -Dacces
s_key_id={your_access_key_id} -Daccess_key={your_access_key_secret}"
    ports:
    - containerPort: 8080
```

> **Note** The following information is displayed:
>
> - `-Dproject`: the name of your project in Log Service. In this example, the name of the project is k8s-log4j.
> - `-DlogStore`: the name of your Logstore in Log Service. In this example, the name of the Logstore is k8s-logstore.
> - `-Dendpoint`: the service endpoint of Log Service. You must configure the service endpoint based on the region where the Log Service project is deployed. For more information, see Service endpoints. In this example, the service endpoint is cn-hangzhou.log.aliyuncs.com.
> - `-Daccess_key_id`: your AccessKey ID.
> - `-Daccess_key`: your AccessKey secret.

iii. Run the following command to create a Deployment:

```
kubectl create -f demo-deployment.yaml
```

iv. Obtain the *demo-Service.yaml* file and run the following command to create a Service:

You do not need to modify the configurations in the *demo-Service.yaml* file.

```
kubectl create -f demo-service.yaml
```

3. Generate Kubernetes cluster logs.

You can run the `kubectl get` command to view the Deployment status of the related resource objects. After the Deployment and Service are deployed, run the `kubectl get svc` command to check the external IP address of the Service, which is EXTERNAL-IP.
Command:

```
kubectl get svc
```

Output:

```
NAME                              TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE
log4j-appender-demo-spring-boot-svc  LoadBalancer  172.21.XX.XX  120.55.XXX.XXX  8080:30398/TCP  1h
```

In this example, you can run the `login` command to generate Kubernetes cluster logs, in which `K8S_SERVICE_IP` is `EXTERNAL-IP`.

> **Note** For a complete list of API operations, see GitHub log4j-appender-demo.

```
curl http://${K8S_SERVICE_IP}:8080/login?name=bruce
```

4. View logs in Log Service
   i. In the **Projects** section of the Log Service console, find and click the k8s-log4j project to go to the details page of the project.
   ii. Click the icon on the right side of the k8s-logstore Logstore and select **Search & Analysis** to view the logs of the Kubernetes cluster.

11.2.4. Collect the logs of control plane components in a managed Kubernetes cluster

Container Service for Kubernetes (ACK) allows you to collect the logs of control plane components in a managed Kubernetes cluster. The logs are collected to a Log Service project that belongs to your account. This topic describes how to enable the log collection feature to collect the logs of control plane components in a managed Kubernetes cluster. It also describes how to view the collected logs.

Prerequisites

Your Alibaba Cloud account has a sufficient quota of Logstores in Log Service.

> **Note** By default, an Alibaba Cloud account can create up to 50 Logstores.

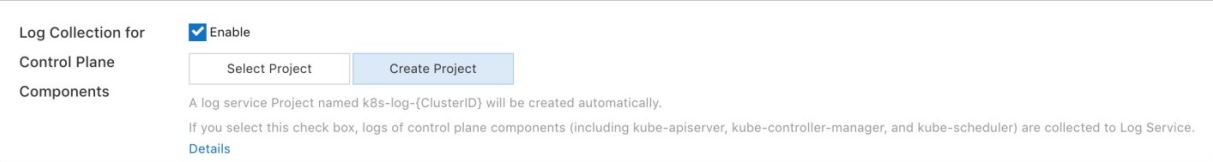
Context

You can manage an ACK cluster in a more secure and effective way by analyzing the logs of the control plane components in the cluster. To enable the log collection feature for control plane components in a standard or professional managed Kubernetes cluster, select **Collect Logs of Control Plane Components** when you create the cluster. The logs are shipped to the specified Log Service project that belongs to your account as log streams. You are charged based on the billing rules of Log Service on a pay-as-you-go basis. For more information, see Pay-as-you-go.

Enable log collection for control plane components

When you create a cluster, select **Enable** for **Log Collection for Control Plane Components** on the **Component Configurations** wizard page. For more information, see 创建Kubernetes托管版集群 or Create a professional managed Kubernetes cluster.

> **Note**
>
> - By default, **Enable** is selected when you create a professional managed Kubernetes cluster. By default, this check box is unselected when you create a standard managed Kubernetes cluster.
> - You can select an existing Log Service project in the **Log Collection for Control Plane Components** section.



View the logs of control plane components

After you create the managed Kubernetes cluster, you can view the logs of the control plane components by using one of the following methods:

Method 1: View the logs of control plane components in the Log Service console.

1. Log on to the Log Service console.
2. In the **Projects** section, click the name of the Log Service project that is used for the cluster.
3. On the **Log Storage** page, click the **Logstores** tab on the left side of the page and click the Logstore where the logs of control plane components are stored. You can query the logs of the following components: **kube-apiserver**, **kube-scheduler**, and **kube-controller-manager**. For more information, see Overview of Log Service.

Method 2: View the logs of control plane components in the ACK console.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. View the three control plane components.
   - You can view the three control plane components through the **Cluster Information** menu.
     a. On the cluster details page, click the **Cluster Resources** tab and click the URL of the Log Service project.
     b. On the **Log Storage** page, click the **Logstores** tab on the left side of the page and click the Logstore where the logs of control plane components are stored. You can query the logs of the following components: **kube-apiserver**, **kube-scheduler**, and **kube-controller-manager**. For more information, see Overview of Log Service.
   - You can also view the three control plane components through the **Operations** menu.
     a. In the left-side navigation pane of the cluster details page, choose **Operations > Log Center**.
     b. Click the **Logs of Control Plane Components** tab. You can select a component and view its logs.

Logstores for control plane components

ACK allows you to collect the logs of the following control plane components. The logs of each component are stored in a separate Logstore. For more information about the components, see Kubernetes components.

Component	Logstore	Description
kube-apiserver	apiserver	kube-apiserver is a component of the Kubernetes control plane that exposes the Kubernetes API. For more information, see kube-apiserver.
kube-controller-manager	kcm	kube-controller-manager is the control center of a Kubernetes cluster and runs controller processes. For more information, see kube-controller-manager.
kube-scheduler	scheduler	kube-scheduler is the default scheduler of a Kubernetes cluster. For more information, see kube-scheduler.

Related information

- Pay-as-you-go
- Log search
- 创建Kubernetes托管版集群
- Create a professional managed Kubernetes cluster

11.2.5. Monitor and analyze the log of CoreDNS

CoreDNS is deployed in Container Service for Kubernetes (ACK) clusters and serves as a DNS server. You can check the log of CoreDNS to analyze the reasons why the DNS resolution is slow or resolution queries for high-risk domain names are received. This topic describes how to enable the logging and monitoring of CoreDNS.

Prerequisites

- The logging component alibaba-log-controller is installed.
  By default, alibaba-log-controller is installed when a cluster is created. If alibaba-log-controller is not installed, you can manually install the component. For more information, see Collect log files from containers by using Log Service.
- Make sure that the version of alibaba-log-controller is 0.2.0.0-76648ee-aliyun or later. For more information about how to upgrade a component, see Manage system components.

Step 1: Enable the logging of CoreDNS

> **Note** After you enable the logging of CoreDNS, the CPU usage increases by about 10% and the data transfer also increases. If the pod replicas of CoreDNS are running with high CPU usage, we recommend that you scale out CoreDNS. For more information about how to scale out CoreDNS, see Manually scale the number of pods for an application.

ACK creates a ConfigMap named **coredns** in the kube-system namespace of a cluster. You can modify the **coredns** ConfigMap by specifying the logging component in the log field of the Corefile configuration. This enables the logging of CoreDNS. For more information about how to modify a ConfigMap, see Modify a ConfigMap.

The following content is an example of the **coredns** ConfigMap that uses the default log format:

```
Corefile: |
 .:53 {
    errors
    log // Specify the logging component.
    health {
      lameduck 5s
    }
    ready
    kubernetes cluster.local in-addr.arpa ip6.arpa {
     pods insecure
     upstream
     fallthrough in-addr.arpa ip6.arpa
     ttl 30
    }
    prometheus :9153
    forward . /etc/resolv.conf
    cache 30
    loop
    reload
    loadbalance
  }
  // If you want to log resolution queries of containers in other domains, you must specify the logging component
for these domains by using the same configuration format.
  demo.com:53 {
    ...
    log // Specify the logging component.
  }
```

Step 2: Configure the logging of CoreDNS

Logging configurations can be implemented by using CustomResourceDefinitions (CRDs). You can create custom resource objects of the AliyunLogConfig type. alibaba-log-controller automatically configures Log Service settings and creates log reports based on the created custom resource objects. For more information about how to create a CRD to define the AliyunLogConfig custom resource object, see Manage custom resources.

The following YAML template is a sample AliyunLogConfig configuration:

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
 #    Your config name, must be unique in you k8s cluster.
 name: k8s-coredns-log
 namespace: kube-system
spec:
 # logstore name to upload log
 logstore: coredns-log
 # logtail config detail
 productCode: k8s-coredns
 logtailConfig:
  inputType: plugin
  # logtail config name, should be same with [metadata.name]
  configName: k8s-coredns-log
  inputDetail:
   plugin:
    inputs:
    - type: service_docker_stdout
```

```
    detail:
      IncludeLabel:
        io.kubernetes.container.name: coredns
      Stderr: true
      Stdout: true
    processors:
    - type: processor_regex
      detail:
        KeepSource: false
        KeepSourceIfParseError: true
        Keys:
        - level
        - remote
        - port
        - id
        - type
        - class
        - name
        - proto
        - size
        - do
        - bufsize
        - rcode
        - rflags
        - rsize
        - duration
        NoKeyError: true
        NoMatchError: false
        FullMatch: false
        Regex: \[([^]]+)]\s([^:]+):(\S+)\s+-\s+(\S+)\s+"(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+([^"]+)"\s+(\S+)\s+(
\S+)\s+(\S+)\s+([\d\.]+).*
        SourceKey: content
    - type: processor_regex
      detail:
        KeepSource: false
        KeepSourceIfParseError: true
        Keys:
        - error
        - rcode
        - name
        - type
        - errorMsg
        NoKeyError: false
        NoMatchError: false
        FullMatch: false
        Regex: \[ERROR]\s+(plugin/errors):\s+(\S)+\s+(\S+)\s+([^:]*):\s+(.*)
        SourceKey: content
```

> **Notice**
>
> - Make sure that the version of alibaba-log-controller is 0.2.0.0-76648ee-aliyun or later. If the CRD that defines the AliyunLogConfig type object is already created, delete and recreate it before you upgrade alibaba-log-controller.
> - The preceding configurations take effect for only the default log format of CoreDNS. If you use a custom log format for CoreDNS, modify the regular expression in the Regex field.
>   For more information about how to customize the log format of CoreDNS, see log.
>   For more information about the logging and logging configurations, see Use CRDs to collect Kubernetes container logs in DaemonSet mode.

Step 3: Check the log of CoreDNS

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the details page of the cluster, click the **Cluster Resources** tab and click the hyperlink on the right side of **Log Service Project**.
5. In the left-side navigation page of the Logstores page, click and click **Dashboard**. In the Dashboard list, find and click **Kubernetes CoreDNS Log Analysis**
   On the **Kubernetes CoreDNS Log Analysis** page, you can view aggregated information about the number of requests to CoreDNS, the success rate of resolution requests, and the request latencies. You can also view the list of most frequently accessed domain names, the list of invalid domain names, the list of slow resolutions, and the list of access to high-risk domain names.

Step 4: Configure alerts based on the log of CoreDNS

On the **Kubernetes CoreDNS Log Analysis** page, you can configure alerts based on each chart or list.

1. Find the chart or list that you want to use and choose > **Save as Alert** in the upper-right corner. For more information about how to configure an alert, see Create an alert rule.

   After an alert rule is created, you can manage the alert rule and view the alerts that are triggered based on the alert rule. For more information, see Manage an alert rule and View alerts.

Related information

- Overview
- Introduction and configuration of the DNS service in ACK clusters

11.3. Monitoring management

11.3.1. Monitor basic resources

Resource monitoring is one of the most commonly used monitoring methods in Kubernetes. You can use resource monitoring to check the resource usage of workloads. The resources include CPU, memory, and network resources. Container Service for Kubernetes (ACK) integrates Cloud Monitor to provide resource monitoring features. By default, ACK installs the Cloud Monitor agent for new clusters. This topic describes how to monitor basic resources and configure alerts by using the ACK console.

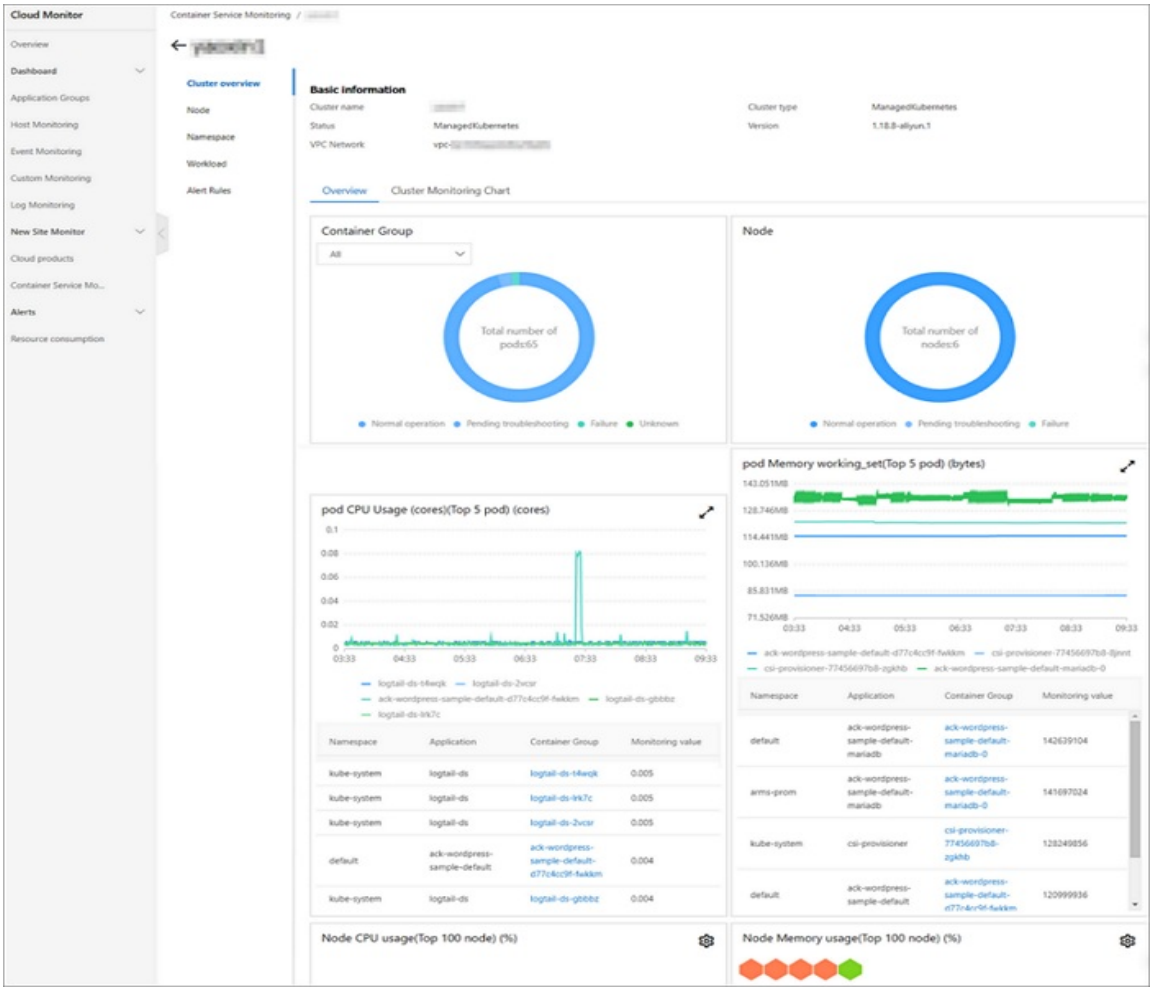
Prerequisites

To use the latest Cloud Monitor version, the metrics-server component must be upgraded to V0.3.8.5 or

later. For more information, see Install the metrics-server component.

Features

- Provides comprehensive metrics to help you gain insight into cluster performance.



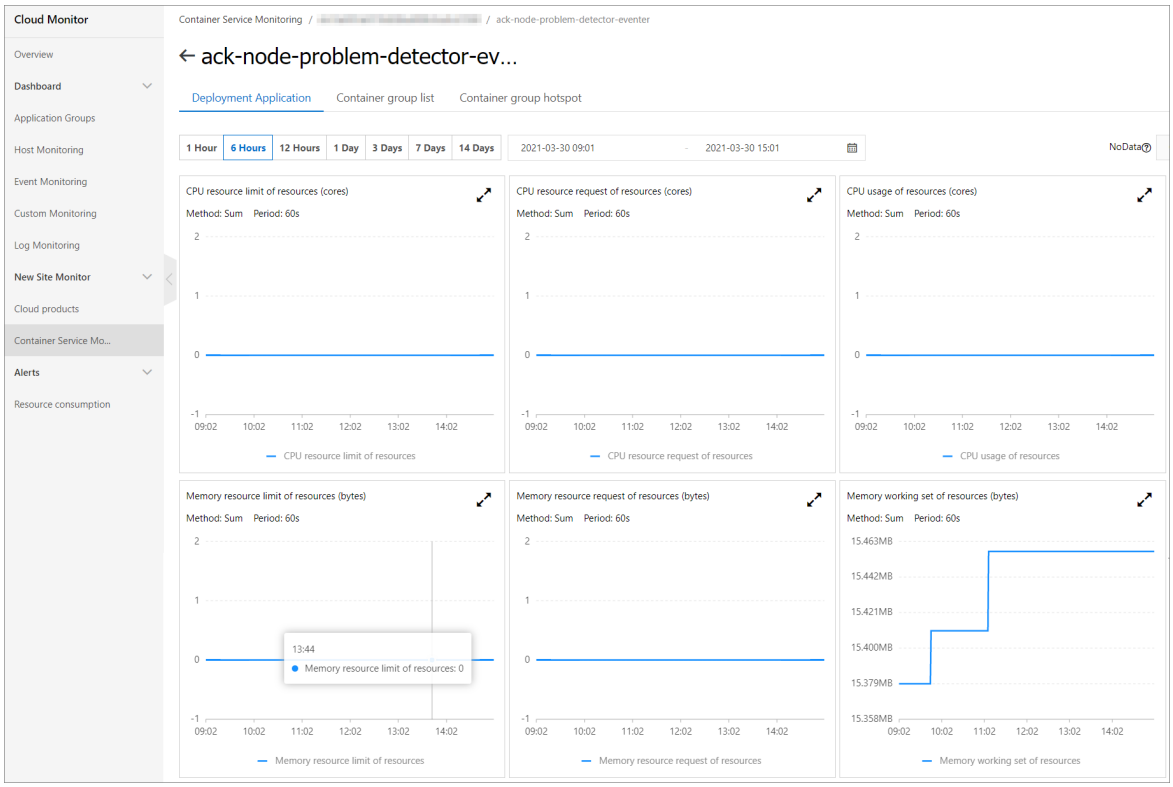
- Improves monitoring and alerting capabilities.
  Upgrades Cloud Monitor to the latest version to provide professional capabilities of container resource monitoring. Provides monitoring metrics for native Kubernetes objects, such as namespaces, nodes, workloads, and pods. Upgrades the alerting feature and allows you to configure alert rules based on different perspectives.
- Provides appropriate metrics for different monitoring scenarios.
  Supports the most appropriate metrics for different scenarios, such as the host infrastructure layer, container layer in Platform as a Service (PaaS), and Kubernetes scheduling layer. For example, the memory metrics that affect Kubernetes scheduling in containers are dedicated to the working memory of containers. This helps distinguish container memory usage from host memory usage.

Go to Resource Monitoring

Method 1: Go to Resource Monitoring by using the ACK console

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

5. On the **Deployments** page, find the application that you want to monitor and click **Monitor** in the **Actions** column to go to the **Container Service Monitoring** page.



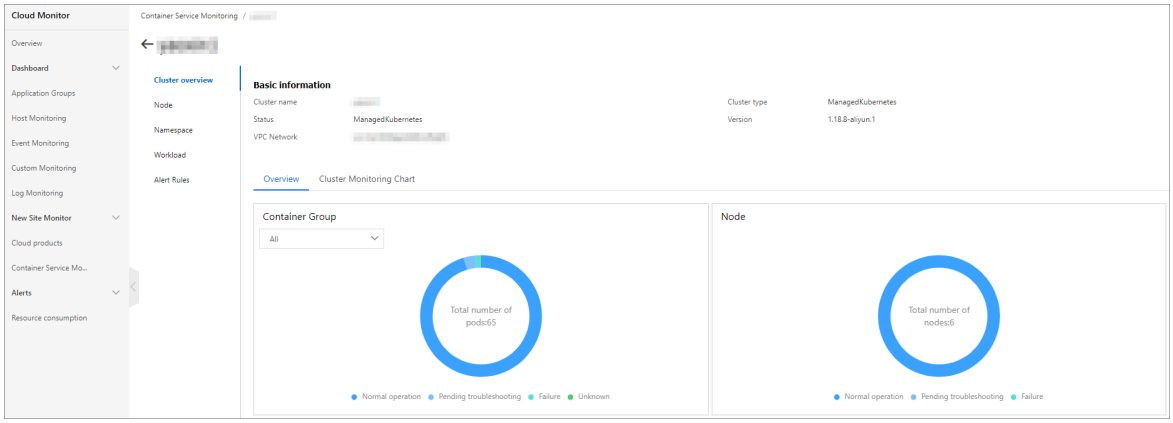
6. You can click the **Deployment Application**, **Container group list**, and **Container group hotspot** tabs to view monitoring data.

Method 2: Go to Resource Monitoring by using the Cloud Monitor console

1. Log on to the Cloud Monitor console.
2. In the left-side navigation pane, click **Container Service Monitoring**.
3. On the **Container Service Monitoring** page, find the cluster that you want to monitor and click the name of the cluster or click **View the Detail** in the **Actions** column.

> **Note** The first time you visit the page, a message appears and requires you to perform authorization. You must click **Authorize** to complete authorization before you can go to the details page.

4. On the page that appears, you can click **Cluster overview**, **Node**, **Namespace**, and **Workload** to view application monitoring data from different perspectives.



Configure alerts based on scenarios

Scenario	Description	How to configure
Monitor the health status of the cluster and send alerts on resource usage exceptions in the cluster or nodes.	When resource usage exceptions occur in the cluster or nodes, alerts need to be sent at the earliest opportunity to prevent service interruptions. We recommend that you configure alert rules to monitor the resource usage of the entire cluster or all nodes in the cluster.	When you create an alert rule, set **Resource Range** to **Cluster** or **Node**. This allows you to detect abnormal metrics in the entire cluster or any node in the cluster. If you set Resource Range to Node, make sure that you select **All** nodes. This triggers an alert when an abnormal value of the metric specified in Rule Description is detected in any node in the cluster.
Monitor the resource usage of pods and send alerts on any pod in the cluster.	When a resource usage exception occurs in the cluster, the exception need to be analyzed to find the pod that causes the problem. We recommend that you configure alert rules to monitor the resource usage of all pods in the cluster.	When you create an alert rule, set **Resource Range** to **Container Group (pod)** and set both Namespace and Container Group (pod) to **All**. This triggers an alert when an abnormal value of the metric specified in Rule Description is detected in any pod in the cluster.
Monitor the cluster by namespace and send alerts on pods in a specified namespace in the cluster.	In most cases, a cluster is shared among multiple applications. Namespaces provide a commonly used method to isolate applications in a multi-tenant environment. When a resource usage exception occurs in an application of a specified namespace, alerts need to be sent at the earliest opportunity. We recommend that you configure alert rules to monitor the resource usage of all pods in a specified namespace in the cluster.	When you create an alert rule, set **Resource Range** to **Container Group (pod)**, set **Namespace** to the one where your application belongs, and set **Container Group (pod)** to **All**. This triggers an alert when an abnormal value of the metric specified in Rule Description is detected in any pod in the specified namespace.

Scenario	Description	How to configure
Monitor the resource usage of applications and send alerts on pods of a specified workload in a specified namespace.	In most cases, a cluster is shared among multiple applications. Workloads provide a commonly used method to isolate applications in a multi-tenant environment. For example, an application may be run as a Deployment. When a resource usage exception occurs in a Deployment of a specified application, alerts need to be sent at the earliest opportunity. We recommend that you configure alert rules to monitor the resource usage of all pods of a specified workload.	When you create an alert rule, set **Resource Range** to **Container Group (pod)**, set **Namespace** to the one where your application belongs, and select the workload type of your application. The following types of workload are supported: Deployment, StatefulSet, DaemonSet, Job, and CronJob. Set **Container Group (pod)** to All. This triggers an alert when an abnormal value of the metric specified in Rule Description is detected in any pod of the specified workload.

Configure alert rules

Step 1: Create an alert contact and add it to an alert contact group

1. Log on to the Cloud Monitor console.
2. In the left-side navigation pane, choose **Alerts > Alert Contacts**.
3. Create an alert contact and add it to an alert contact group.

   For more information, see Create an alert contact or alert group.

Step 2: Create an alert rule

1. Log on to the Cloud Monitor console.
2. In the left-side navigation pane, click **Container Service Monitoring**.
3. On the **Container Service Monitoring** page, select the cluster for which you want to create an alert rule and click **View Alert Rules** in the **Actions** column.
4. On the **Alert Rules** page, click **Create Alert Rule**.
5. In the **Create Alert Rule** panel, configure the parameters.

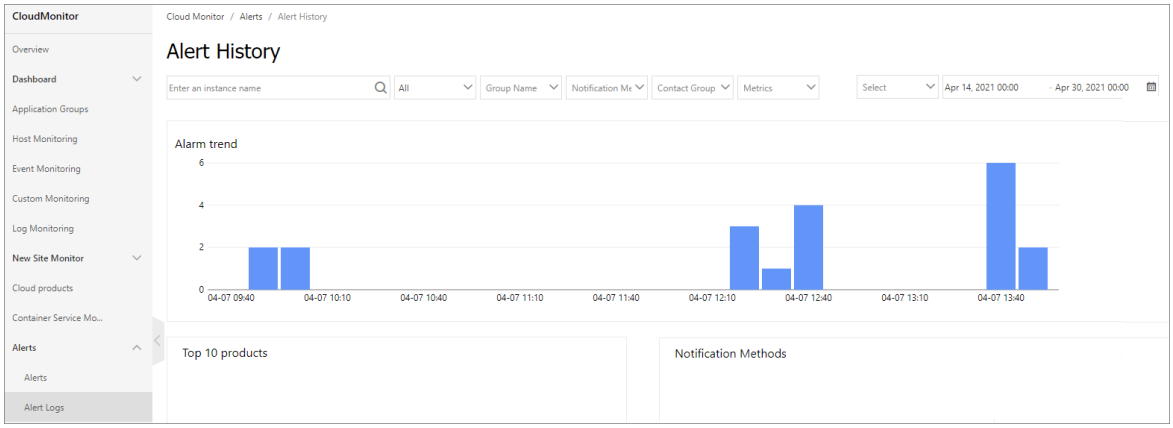
Parameter	Description
Resource Range	The resources to which the alert rule is applied. Valid values: ○ **Cluster**: The alert rule is applied to the cluster. ○ **Node**: The alert rule is applied to all nodes or specified nodes in the cluster. ○ **Container Group (pod)**: The alert rule is applied to all pods or specified pods in the specified application under the specified namespace of the cluster. If you select **All**, an alert is triggered when the metric of any pod exceeds the threshold specified in Rule Description.
Rule Description	The content of the alert rule. Configure the metric, threshold, and alert level. For more information about pod metrics, see ACK (new version). The parameters in this section specify the conditions that trigger an alert.
Effective Time	The interval at which new alert notifications are sent if the alert is not cleared.
Effective From	The time period during which the alert rule is effective. Cloud Monitor checks whether the monitoring data meets the alert rule only during the effective period.

Parameter	Description
HTTP Callback	Cloud Monitor sends a POST request to push an alert to the specified callback URL. Only HTTP requests are supported. **Note** We recommend that you specify a callback URL that can be accessed over the Internet.
Alert Contact Group	The alert contact group that receives alert notifications.

6. Click **OK** to create the alert rule.
   On the **Alert Rules** page, verify that the new alert rule is displayed.

Verification

1. In the left-side navigation pane, choose **Alerts > Alert Logs**.
2. On the **Alert History** page, you can view alert trends and details of alert history.



Previous Resource Monitoring page

If the metrics-server component of your cluster is not upgraded to V0.3.8.5 or later, you can perform the following steps to go to the previous Resource Monitoring page:

1. Log on to the ACK console.
2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
4. Select the Deployment that you want to monitor and click **Monitor** in the Actions column. Then, you are redirected to the **Dashboards** page in the Cloud Monitor console.
5. You can click the **Deployment Application**, **Container group list**, and **Container group hotspot** tabs to view monitoring data.
6. (Optional)To configure alerts, choose **Alerts > Alert Rules** in the left-side navigation pane.

   The name of a group-based metric starts with group and the name of an instance-based metric starts with pod. For more information, see Manage alert rules.

11.3.2. Monitor application performance

Container Service for Kubernetes (ACK) allows you to use Application Real-Time Monitoring Service (ARMS) to monitor Java and PHP applications that are deployed in clusters. ARMS can automatically discover application topologies, generate 3D topologies, discover and monitor API endpoints, and detect abnormal and slow transactions. ARMS provides an efficient method to diagnose and troubleshoot application issues.

Prerequisites

- Create a cluster
- Activate and upgrade ARMS

> **Note** The PHP application monitoring feature is in public preview and free of charge.

Context

ARMS is an application performance management (APM) service of Alibaba Cloud. After you install the ARMS application monitoring agent in an ACK cluster, you can use ARMS to monitor Java applications in the cluster without code modifications. ARMS allows you to quickly locate abnormal and slow transactions, reproduce the parameters of API calls, detect memory leaks, and discover system bottlenecks. This significantly improves the efficiency of diagnosing and troubleshooting application issues. For more information, see Overview.

Install the ARMS application monitoring agent

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**. On the **Alibaba Cloud Apps** tab, find and click **ack-arms-pilot**.
3. On the **App Catalog - ack-arms-pilot** page, select the cluster that you want to monitor in the **Deploy** section and click **Create**.

Authorize ACK to access ARMS

Perform the following steps to grant ACK the permissions to access ARMS.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the details page of the cluster, click the **Cluster Resources** tab and then click the name of the Resource Access Management (RAM) role of worker nodes.
5. You are redirected to the **RAM Roles** page in the **RAM** console. On the RAM Roles page, click the policy name on the **Permissions** tab.
6. On the **Policy Document** tab, click **Modify Policy Document**. In the Modify Policy Document panel, copy the following content to the Policy Document field and click **OK**.

```
{
  "Action": "arms:*",
  "Resource": "*",
  "Effect": "Allow"
  }
```

Enable ARMS to monitor Java applications

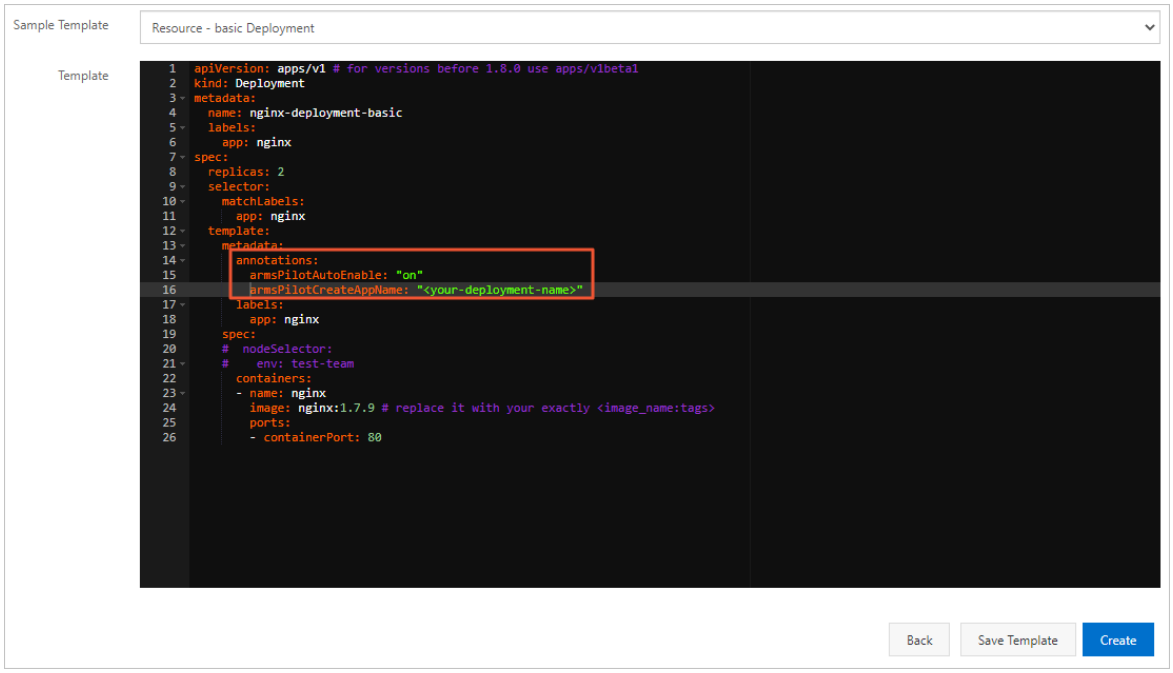
The following steps describe how to enable ARMS to monitor newly created Java applications and existing Java applications.

To enable ARMS when you create an application, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. On the **Create** page, select a template from the **Sample Template** drop-down list and add the following `annotations` to the *spec/template/metadata* section in **Template**.

> **Note** Replace *<your-deployment-name>* with the name of your application.

```
annotations:
 armsPilotAutoEnable: "on"
 armsPilotCreateAppName: "<your-deployment-name>"
```



The following YAML template shows how to create a stateless application and enable ARMS for the application:

Show the complete YAML file (Java) ▼

Enable ARMS to monitor PHP applications

The following steps describe how to enable ARMS to monitor newly created PHP applications and existing PHP applications.

To enable ARMS when you create an application, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

5. On the **Deployments** tab, click **Create from Template** in the upper-right corner.
6. On the **Create** page, select a template from the **Sample Template** drop-down list and add the following `annotations` to the *spec/template/metadata* section in **Template**.

   > **Note** Replace *<your-deployment-name>* with the name of your application.

   ```
   annotations:
    armsPilotAutoEnable: "on"
    armsPilotCreateAppName: "<your-deployment-name>"
    armsAppType: PHP
   ```

7. If you install the ARMS application monitoring agent for the first time, you must modify the *arms-php.ini* ConfigMap in the arms-pilot namespace. The content of the file is the same as that of the *php.ini* file of the PHP application.

   > **Note** In the extension=/usr/local/arms/arms-php-agent/arms-7.2.so configuration, *arms-7.2.so* indicates that the version of the PHP application is V7.2. You can change the version value to 5.4, 5.5, 5.6, 7.0, 7.1, or 7.2.

8. Copy the content of the *arms-php.ini* ConfigMap to the *spec/template/spec/containers* section in the *php.ini* file. Set mountPath to the path of the *php.ini* file.

   ```
   volumeMounts:
       - name: php-ini
         mountPath: /etc/php/7.2/fpm/php.ini
         subPath: php.ini
   ```

   ```
   volumes:
      - name: php-ini
        configMap:
          name: arms-php.ini
   ```

   The following YAML template shows how to create a stateless application and enable ARMS for the application:

   Show the complete YAML file (PHP) ▼

To enable ARMS to monitor existing applications, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. Select a namespace from the **Namespace** drop-down list. In the Deployments list, find the application that you want to monitor and then choose **More > View in YAML** in the **Actions** column.
6. In the **Edit YAML** dialog box, add the following `annotations` to the *spec/template/metadata* section and click **Update**.

   > **Note** Replace *<your-deployment-name>* with the name of your application.

```
annotations:
 armsPilotAutoEnable: "on"
 armsPilotCreateAppName: "<your-deployment-name>"
 armsAppType: PHP
```

7. Copy the content of the *arms-php.ini* ConfigMap to the *spec/template/spec/containers* section in the *php.ini* file. Set mountPath to the path of the *php.ini* file.

```
volumeMounts:
    - name: php-ini
      mountPath: /etc/php/7.2/fpm/php.ini
      subPath: php.ini
```

```
volumes:
   - name: php-ini
     configMap:
      name: arms-php.ini
```

Result

On the **Deployments** page or the **StatefulSets** page, find the application and check whether the **ARMS Console** button appears in the **Actions** column.

> **Note** If the **ARMS Console** button does not appear in the **Actions** column, check whether ACK is authorized to access ARMS.

What's next

After you complete the preceding steps, ARMS Alibaba Cloud Container Service for Kubernetes is enabled for the application that is deployed in application monitoring. In the target application's **Action** Column click **ARMS console** . The Application Monitoring page of the ARMS console appears. ARMS application monitoring have the following capabilities:

1. Displays the overall application performance through key metrics. Automatically discovers the application topology.

2. The 3D topology shows the health status of applications, services, and hosts, as well as the upstream and downstream dependencies of the applications. It helps you quickly locate the services that induce faults, the applications affected by the faults, and the associated hosts, and comprehensively diagnose the root cause of the fault.

3. Capture abnormal and slow transactions, and obtain slow SQL statements, MQ accumulation analysis reports, or exception classification reports of the interface. Then, analyze common problems such as right or wrong and slow in more detail.

4. Automatically discovers and monitors common Web frameworks and RPC frameworks in application code, and automatically collects statistics on metrics such as the call volume, response time, and number of errors of Web interfaces and RPC interfaces.

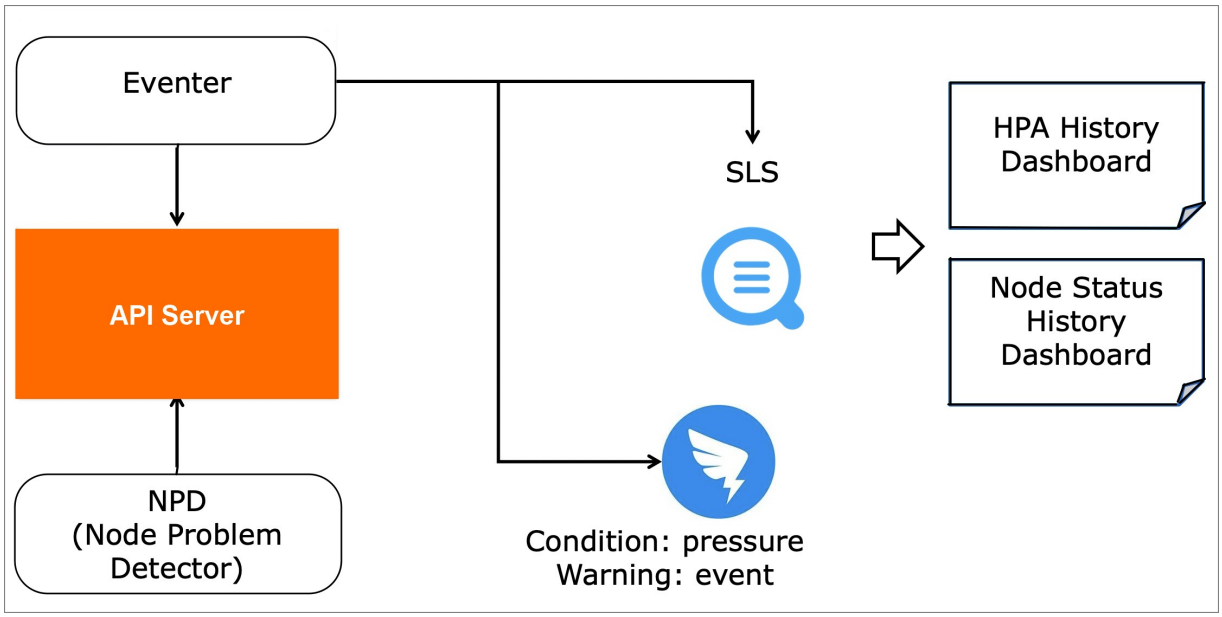
11.3.3. Event monitoring

Event monitoring is a monitoring method provided by Kubernetes. It provides improvements over the resource monitoring in terms of timeliness, accuracy, and scenarios. You can use node-problem-detector with the Kubernetes event center of Log Service to sink cluster events, and configure node-problem-detector to diagnose clusters and send events of anomalies to sinks. You can sink cluster events to DingTalk, Log Service, and EventBridge. This allows you to monitor exceptions and issues in clusters in real time.

Context

Kubernetes is designed based on the state machine. Events are generated due to transitions between different states. Typically, Normal events are generated when the state machine changes to expected states and Warning events are generated when the state machine changes to unexpected states.

Container Service for Kubernetes (ACK) provides out-of-the-box monitoring solutions for events in different scenarios. The node-problem-detector and kube-eventer tools that are maintained by ACK allow you to monitor Kubernetes events.



- node-problem-detector is a tool to diagnose Kubernetes nodes. node-problem-detector detects node anomalies, generates node events, and works with kube-eventer to generate alerts upon these events. node-problem-detector generates node events when the following anomalies are detected: Docker engine hangs, Linux kernel hangs, outbound traffic anomalies, and file descriptor anomalies. For more information, see NPD.
- Kube-eventer is an open source event emitter that is maintained by ACK. Kube-eventer sends Kubernetes events to sinks such as DingTalk, Log Service, and EventBridge. Kube-eventer also provides filter conditions to filter different levels of events. You can use kube-eventer to collect events in real time, trigger alerts upon specific events, and asynchronously archive events. For more information, see kube-eventer.

This topic describes how to configure event monitoring in the following scenarios:

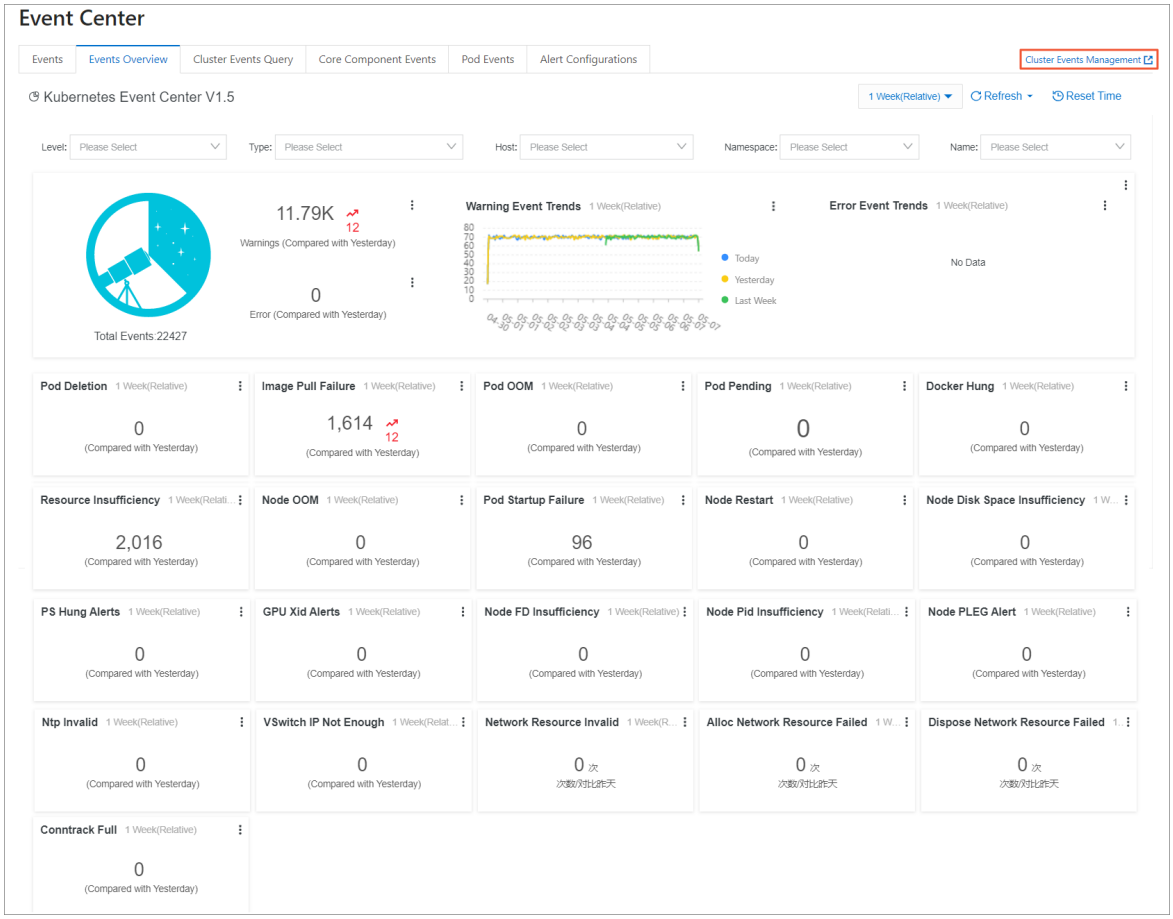
Scenario 1: Use node-problem-detector with the Kubernetes event center of Log Service to sink cluster events

node-problem-detector works with third-party plug-ins to detect node anomalies and generate cluster events. A Kubernetes cluster also generates events when the status of the cluster changes. For example, when a pod is evicted or an image pull fails, a related event is generated. The Kubernetes event center of Log Service collects, stores, and visualizes cluster events. It allows you to query and analyze these events, and configure alerts. To sink cluster events to the Kubernetes event center in the Log Service console, use the following methods:

Method 1: If **Install node-problem-detector and Create Event Center** was selected when you created the cluster, perform the following steps to go to the Kubernetes event center. For more information about how to install node-problem-detector and deploy the Kubernetes event center when you create a cluster, see 创建Kubernetes托管版集群.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. Choose **Operations > Event Center**.

5. Click **Cluster Events Management** in the upper-right corner of the page to go to the K8s Event Center page. In the left-side navigation pane of the **K8s Event Center** page, find the cluster that you want to manage. Then, click the ﹥ icon to the left of the cluster name to view event details that are provided by the Kubernetes event center.
   The Kubernetes event center provides event overview, event details, and information about pod lifecycles. You can also customize queries and configure alerts.



Method 2: If the Kubernetes event center was not deployed when you created the cluster, perform the following steps to deploy and use the Kubernetes event center:

1. Install ack-node-problem-detector in the monitored cluster and enable Log Service for the monitored cluster. For more information, see Scenario 3: Use DingTalk to generate alerts upon Kubernetes events.

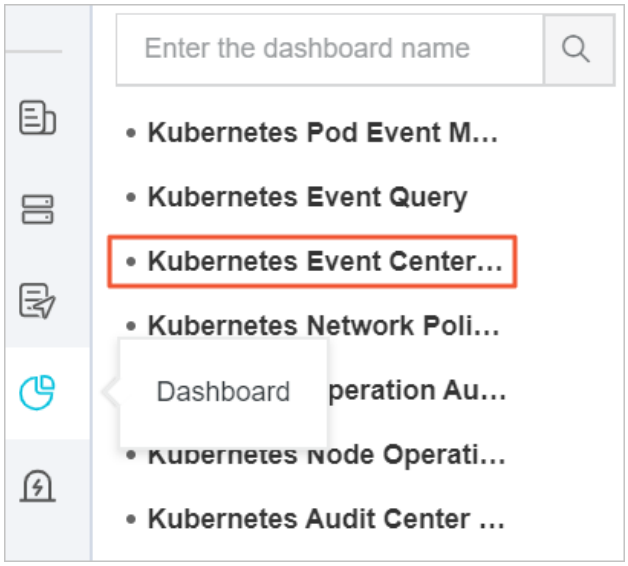
> **Note** If ack-node-problem-detector is deployed but Log Service is disabled, delete and reinstall ack-node-problem-detector.
>
> i. In the left-side navigation pane, click **Clusters**.
> ii. On the **Clusters** page, find the cluster that you want to manage and click the name or click **Details** in the **Actions** column.
> iii. Choose **Applications > Helm**.
> iv. On the **Helm** page, uninstall node-problem-detector by deleting the ack-node-problem-detector plug-in.

When you configure parameters for node-problem-detector, create a Log Service project for the cluster by setting `eventer.sinks.sls.enabled` to true.

```
sinks:
  sls:
   # If you want the monitoring results to be notified by sls, set enabled to true.
    # sls_log_project
   enabled: true
    # You can view the project information by logging in to the
```

After ack-node-problem-detector is redeployed, a related Log Service project is automatically created in the Log Service console for the cluster.

2. Log on to the Log Service console to configure the Kubernetes event center for the cluster.
   i. In the **Import Data** section, click **Kubernetes - Standard Output**.
   ii. Select the Log Service project that is automatically created in the preceding step from the **Project** drop-down list, and select **k8s-event** from the **Logstore** drop-down list.
   iii. Click **Next** and click **Complete Installation**.
3. In the **Projects** section of the Log Service console, find and click the Log Service project.
4. In the left-side navigation pane, click the **Dashboard** icon and click **Kubernetes Event Center V1.1**.



On the dashboard of the Kubernetes event center, you can view all cluster events.

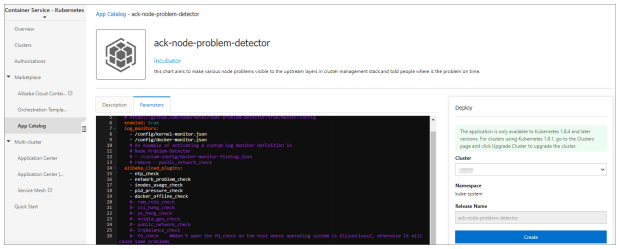
Scenario 2: Configure node-problem-detector to diagnose a cluster and send events of anomalies to sinks

node-problem-detector is a tool that is used to diagnose Kubernetes nodes. node-problem-detector detects node anomalies, generates node events, and works with kube-eventer to generate alerts upon these events. node-problem-detector generates node events when the following anomalies are detected: Docker engine hangs, Linux kernel hangs, outbound traffic anomalies, and file descriptor anomalies. Perform the following steps to configure node-problem-detector:

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**. On the App Catalog page, find and click **ack-node-problem-detector**.

> **Note** If the Kubernetes event center is deployed, you must first delete the ack-node-problem-detector component.
>
> i. In the left-side navigation pane, click **Clusters**.
>
> ii. On the **Clusters** page, find the cluster that you want to manage and click the name or click **Details** in the **Actions** column.
>
> iii. Choose **Applications > Helm**.
>
> iv. On the **Helm** page, delete the ack-node-problem-detector component.

3. On the **App Catalog - ack-node-problem-detector** page, click the **Parameters** tab to view the default configurations of node-problem-detector.



You can set the sink parameters as described in the following table for kube-eventer.
Parameter description

Parameter	Description	Default
npd.image.repository	The image address of node-problem-detector.	registry.aliyuncs.com/acs/node-problem-detector
npd.image.tag	The image version of node-problem-detector.	v0.6.3-28-160499f
alibaba_cloud_plugins	Plug-ins that are used for node diagnostics. For more information, see the Node diagnostics plug-ins supported by node-problem-detector table.	fd_check, ntp_check, network_problem_check, and inode_usage_check are supported.
plugin_settings.check_fd_warning_percentage	The alerting threshold for monitoring the percentage of opened file descriptors.	80
plugin_settings.inode_warning_percenage	The alerting threshold for monitoring the inode usage.	80
eventer.image.repository	The image address of kube-eventer.	registry.cn-hangzhou.aliyuncs.com/acs/eventer
eventer.image.tag	The image version of kube-eventer image.	v1.6.0-4c4c66c-aliyun
eventer.image.pullPolicy	The policy that specifies how the kube-eventer image is pulled.	IfNotPresent

Parameter	Description	Default
eventer.sinks.sls.enabled	Specifies whether to enable Log Service as a sink of kube-eventer.	false
eventer.sinks.sls.project	The name of the Log Service project.	No
eventer.sinks.sls.logstore	The name of the Logstore in the Log Service project.	No
eventer.sinks.dingtalk.enabled	Specifies whether to enable DingTalk as a sink of kube-eventer.	false
eventer.sinks.dingtalk.level	The level of events at which alerts are generated.	warning
eventer.sinks.dingtalk.label	The labels of the events.	No
eventer.sinks.dingtalk.token	The token of the DingTalk chatbot.	No
eventer.sinks.dingtalk.monitorkinds	The type of resource for which event monitoring is enabled.	No
eventer.sinks.dingtalk.monitornamespaces	The namespace of the resources for which event monitoring is enabled.	No
eventer.sinks.eventbridge.enable	Specifies whether to enable eventBridge as a sink of kube-eventer.	false

Node diagnostics plug-ins supported by node-problem-detector are listed in the following table.

Plug-in	Feature	Description
fd_check	Checks whether the percentage of opened file descriptors on each cluster node exceeds 80%.	The default threshold is 80%. The threshold is adjustable. This plug-in requires a great amount of resources. We recommend that you disable this plug-in.
ram_role_check	Checks whether cluster nodes are assigned the required Resource Access Management (RAM) role and whether the AccessKey ID and AccessKey secret are configured for the RAM role.	N/A
ntp_check	Checks whether the system clocks of cluster nodes are properly synchronized through Network Time Protocol (NTP).	The plug-in is enabled by default.
nvidia_gpu_check	Checks whether the NVIDIA GPUs of cluster nodes can generate Xid messages.	N/A

Plug-in	Feature	Description
network_problem_check	Checks whether the connection tracking (conntrack) table usage on each cluster node exceeds 90%.	The plug-in is enabled by default.
inodes_usage_check	Checks whether the inode usage on the system disk of each cluster node exceeds 80%.	The default threshold is 80%. The threshold is adjustable. The plug-in is enabled by default.
csi_hang_check	Checks whether the Container Storage Interface (CSI) plug-in functions as expected on cluster nodes.	N/A
ps_hang_check	Checks whether processes in the uninterruptible sleep (D) state exist in the systems of cluster nodes.	N/A
public_network_check	Checks whether cluster nodes can access the Internet.	N/A
irqbalance_check	Checks whether the irqbalance daemon functions as expected in the systems of cluster nodes.	N/A
pid_pressure_check	Checks whether the ratio of pid processes in the node system to the maximum pid processes allowed in the kernel exceeds 85%.	The plug-in is enabled by default.
docker_offline_check	Checks whether the docker daemon functions as expected on cluster nodes.	The plug-in is enabled by default.

Note Some plug-ins are enabled by default, as shown in the preceding table. You can find these plug-ins if you select **Install node-problem-detector and Create Event Center** when you enable **Log Service** for the cluster. You can also find these plug-ins when you install the **ack-node-problem-detector** component on the **Add-ons** page. You must manually enable some plug-ins when you deploy the **ack-node-problem-detector** component from the **App Catalog** page.

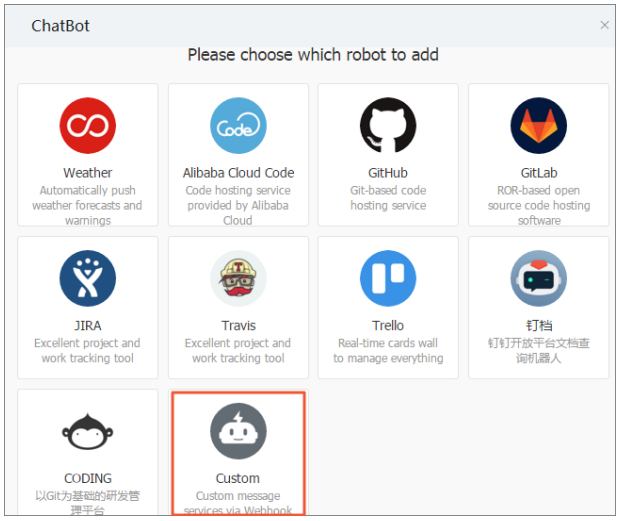
4. In the **Deploy** section, select the monitored cluster from the Cluster drop-down list and click **Create**. The Namespace parameter is automatically set to *kube-system*, and the Release Name parameter is automatically set to *ack-node-problem-detector*.
   Go to the Clusters page. On the Clusters page, find and click the name of the monitored cluster or **Applications** in the **Actions** column. On the page that appears, click the **DaemonSets** tab. On the DaemonSets tab, you can find that **ack-node-problem-detector-daemonset** is running as expected.
   When both node-problem-detector and kube-eventer are running as expected, the system sinks events and generates alerts based on the kube-eventer configuration.

Scenario 3: Use DingTalk to generate alerts upon Kubernetes events

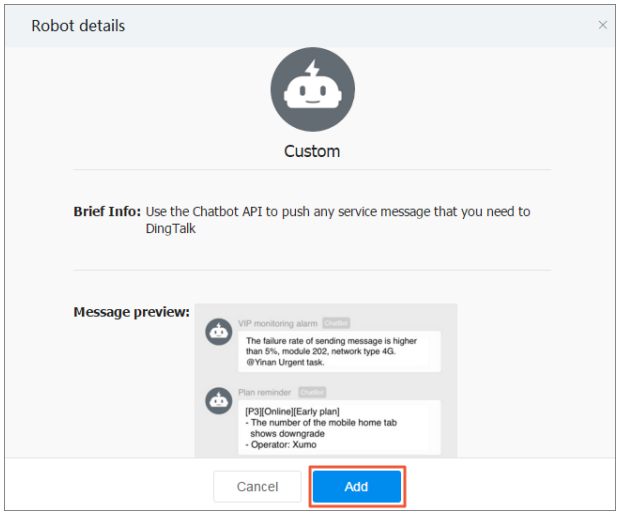
Using a DingTalk chatbot to monitor Kubernetes events and generate alerts is a typical scenario of ChatOps. Perform the following steps to configure node-problem-detector:

1. Click the
   
   icon in the upper-right corner of the chatbox of a DingTalk group to open the **Group Settings** page.
2. Click **Group Assistant**, and then click **Add Robot**. In the ChatBot dialog box, click the **+** icon and select

the chatbot that you want to use. In this example, **Custom** is selected.



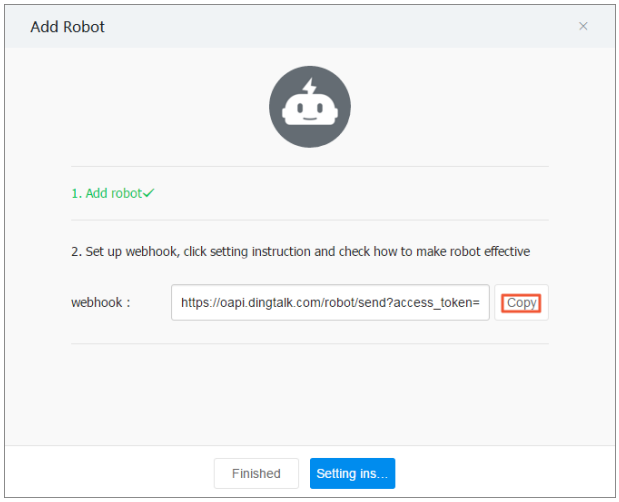
3. On the **Robot details** page, click **Add** to open the **Add Robot** page.



4. Set the following parameters, read and accept the DingTalk Custom Robot Service Terms of Service, and then click **Finished**.

Parameter	Description
Edit profile picture	The avatar of the chatbot. This parameter is optional.
Chatbot name	The name of the chatbot.
Add to Group	The DingTalk group to which the chatbot is added.
Security settings	Three types of security setting are supported: custom keywords, additional signatures, and IP addresses (or CIDR blocks). Only **Custom Keywords** are supported for filtering alerts that are generated upon cluster events. Select **Custom Keywords** and enter *Warning* to receive alerts. If the chatbot frequently sends messages, you can add more keywords to filter the messages. You can add up to 10 keywords. Messages from ACK are also filtered through these keywords before the chatbot sends them to the DingTalk group.

5. Click **Copy** to copy the webhook URL.



> **Note** On the **ChatBot** page, find the chatbot and click the
>
> 
>
> icon to perform the following operations:
>
> - Modify the avatar and name of the chatbot.
> - **Enable** or **disable** message push.
> - Reset the webhook URL.
> - Remove the chatbot.

6. Log on to the ACK console.
7. In the left-side navigation pane, choose **Marketplace > App Catalog**. On the **App Catalog** page, find and click **ack-node-problem-detector**.

> **Note** If the Kubernetes event center is deployed, you must first delete the ack-node-problem-detector component.
>
> i. In the left-side navigation pane, click **Clusters**.
> ii. On the **Clusters** page, find the cluster that you want to manage and click the name or click **Details** in the **Actions** column.
> iii. Choose **Applications > Helm**.
> iv. On the **Helm** page, delete the ack-node-problem-detector component.

8. On the **App Catalog - ack-node-problem-detector** page, click the **Parameters** tab and modify the following settings:
   - In the `npd` section, set the `enabled` parameter to false.
   - In the `eventer.sinks.dingtalk.enabled` section, set the enabled parameter to true.
   - Enter the token that is contained in the webhook URL generated in Step 5.

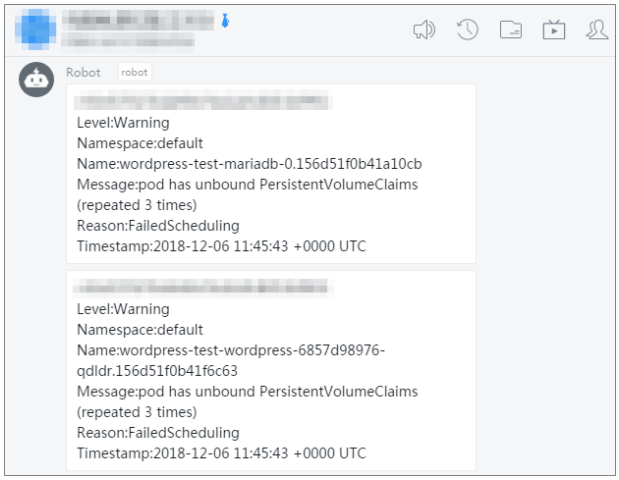
```
eventer:
  image:
    repository: registry-vpc.cn-beijing.aliyuncs.com/acs/kube-eventer-amd64
    tag: "v1.2.2-1be989d-aliyun"
    pullPolicy: IfNotPresent
  # Set environment variables.
  env:
  - TZ: Asia/Shanghai
  #- AccessKeyId: ""
  #- AccessKeySecret: ""
  #- RegionId: ""

  sinks:
    sls:
      # If you want the monitoring results to be notified by sls, set enabled to true.
      # sls log project
      enabled: true
      # You can view the project information by logging in to the
      # SLS console. Please fill in the name of the project here.
      # eg: your project name is k8s-log-cc18a5f3443dhdss22654da,
      # then you can fill k8s-log-cc18a5f3443dhdss22654da to project label.
      project:  "k8s-log-{YOUR_CLUSTER_ID}"
      # if you want to use event center,logstore must be k8s-event. so You'd better not change this value.
      logstore: "k8s-event"
      # topic of sls logstore(optional,default:"")
      topic: ""
      #Whether it is an intranet domain name
      internal: true
      # You can view the project information by logging in to the
      # SLS console. Please fill the logstore address in here.
```

9. On the right side of the page, select the cluster for which you want to enable event monitoring. Then, verify that the Namespace parameter is set to kube-system and the Release Name parameter is set to ack-node-problem-detector, and click **Create**.

Expected results:

Kube-eventer takes effect about 30 seconds after the deployment is complete. When an event with a severity level higher than the threshold occurs, you will receive an alert in the DingTalk group, as shown in the following figure.



Scenario 4: Sink Kubernetes events to Log Service

You can sink Kubernetes events to Log Service for persistent storage, and archive and audit these events. For more information, see Create and use a Kubernetes event center.

1. Create a Log Service project and a Logstore.
   i. Log on to the Log Service console.
   ii. In the **Projects** section, click **Create Project**. In the Create Project panel, set the parameters and click **OK**.

   In this example, a Log Service project named k8s-log4j is created in the China (Hangzhou) region where the monitored ACK cluster is deployed.

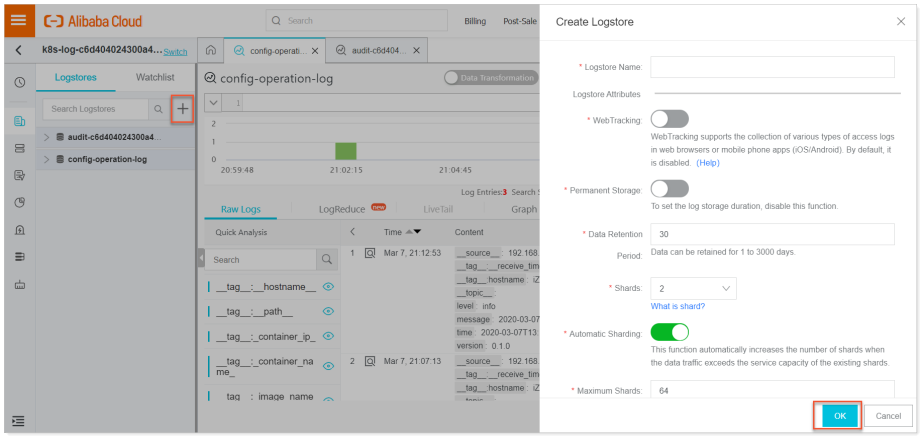
   > **Note** We recommend that you create a Log Service project in the region where the monitored ACK cluster is deployed. When a Log Service project and an ACK cluster are deployed in the same region, log data is transmitted over the internal network. This enables quick retrieval of log data. This also avoids cross-region transmission, which requires additional bandwidth and time costs.

   iii. In the Projects section, find and click the k8s-log4j project. The details page of the project appears.

iv. On the **Logstores** tab, click the **+** icon to open the **Create Logstore** panel.

v. In the Create Logstore panel, set the parameters and click **OK**.

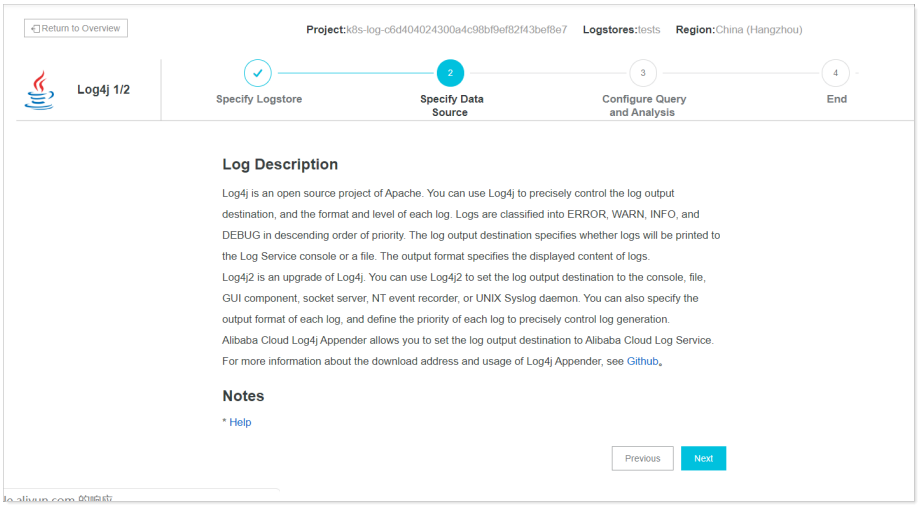
In this example, a Logstore named k8s-logstore is created.



vi. After the k8s-logstore Logstore is created, instructions on how to use the Data Import wizard appear on the page. Click **Data Import Wizard**. The **Import Data** dialog box appears.

vii. Select **log4jAppender** and configure the settings by following the steps on the page.

In this example, Log4jAppender uses the default settings. You can also customize the settings to meet your business requirements.



2. Configure Log4jAppender for the monitored ACK cluster.

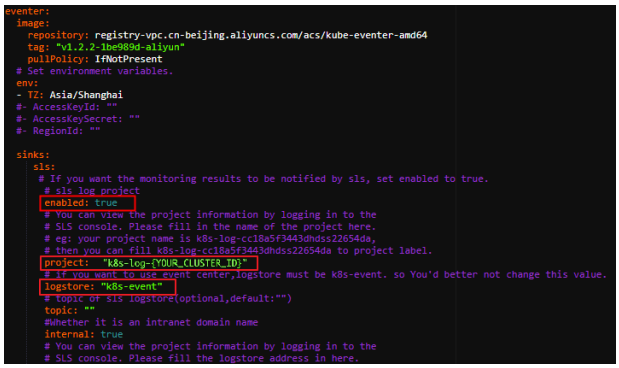
i. Log on to the ACK console.

ii. In the left-side navigation pane, choose **Marketplace > App Catalog**. On the App Catalog page, find and click **ack-node-problem-detector**.

> **Note** If the Kubernetes event center is deployed, you must first delete the ack-node-problem-detector component.
>
> a. In the left-side navigation pane, click **Clusters**.
>
> b. On the **Clusters** page, find the cluster that you want to manage and click the name or click **Details** in the **Actions** column.
>
> c. Choose **Applications > Helm**.
>
> d. On the **Helm** page, delete the ack-node-problem-detector component.

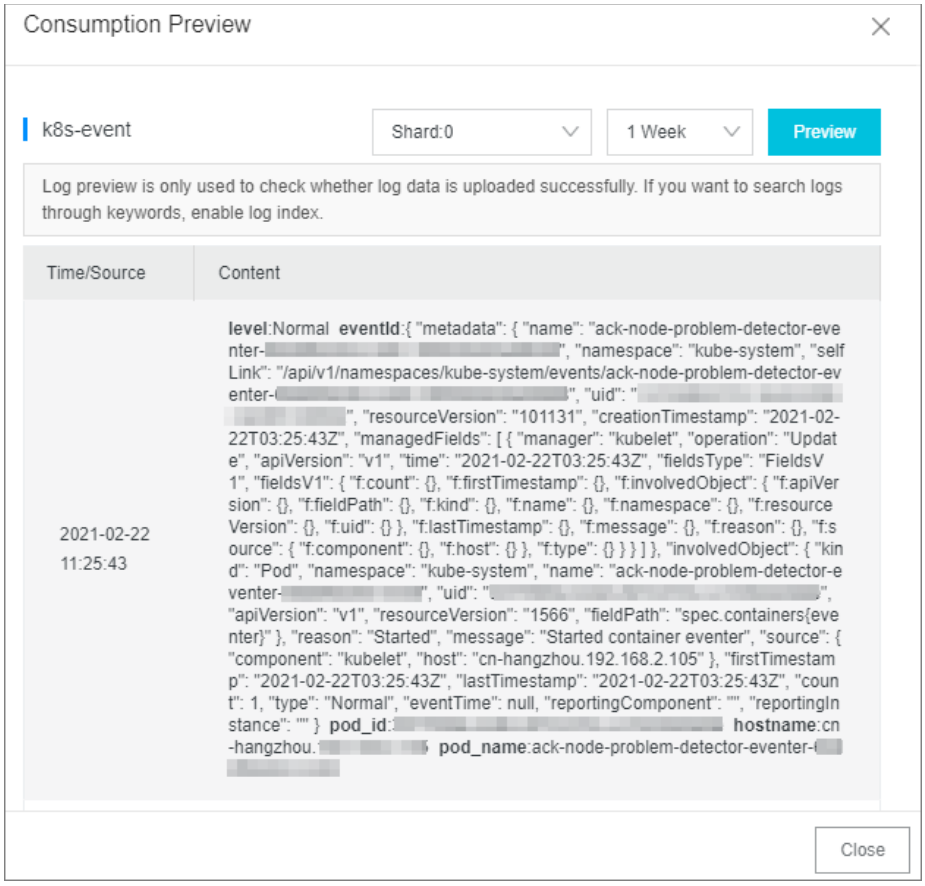
iii. On the **App Catalog - ack-node-problem-detector** page, click the **Parameters** tab and modify the following settings:

- In the `npd` section, set the `enabled` parameter to false.
- In the `eventer.sinks.dingtalk.enabled` section, set the enabled parameter to true.
- Enter the names of `Project` and `Logstore` that are created in Step 1 in the related fields. If you do not customize `Project` when you create the ACK cluster, the `Project` parameter is set to k8s-log-{YOUR_CLUSTER_ID} by default.



iv. Click **Create** to deploy kube-eventer in the monitored cluster.

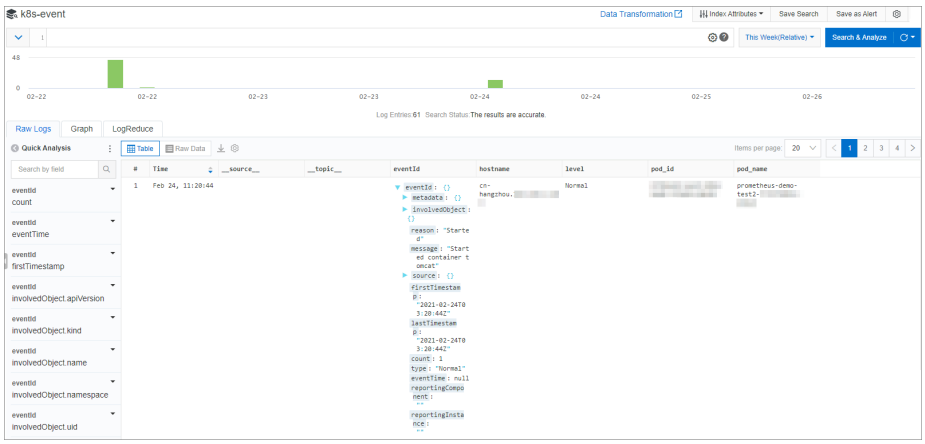
3. An event is generated after an operation is performed on the cluster, such as a pod deletion or an application creation. You can log on to the Log Service console to view the collected log data. For more information, see Consume log data.



4. Set indexes and archiving. For more information, see Configure indexes.
   i. Log on to the Log Service console. In the **Projects** section, find and click the name of the project.
   ii. Click the icon next to the name of the Logstore, and then select **Search & Analysis**.
   iii. In the upper-right corner of the page that appears, click **Enable Index**.
   iv. In the **Search & Analysis** panel, set the parameters.

v. Click **OK**.

The log query and analysis page appears.



> **Note**
>
> - The index configuration takes effect within one minute.
> - A newly enabled or modified index applies to only data that is imported after the index is enabled or modified.

vi. If you need to implement offline archiving and computing, you can ship data from the Logstore to **Object Storage Service (OSS)**. For more information, see Ship log data to OSS.

Scenario 5: Sink Kubernetes events to EventBridge

EventBridge is a serverless event service provided by Alibaba Cloud. Alibaba Cloud services, custom applications, and software as a service (SaaS) applications can connect to EventBridge in a standardized and centralized manner. In addition, EventBridge can route events among these applications based on standardized CloudEvents 1.0 protocol. ACK events can be sunk to EventBridge, which allows you to build a loosely-coupled and distributed event-driven architecture in EventBridge. For more information about EventBridge, see What is EventBridge?.

1. Activate EventBridge. For more information, see Activate EventBridge and grant permissions to a RAM user.
2. Log on to the ACK console.
3. In the left-side navigation pane, choose **Marketplace > App Catalog**. On the App Catalog page, find and click **ack-node-problem-detector**.

> **Note** If the Kubernetes event center is deployed, you must first delete the ack-node-problem-detector component.
>
> i. In the left-side navigation pane, click **Clusters**.
>
> ii. On the **Clusters** page, find the cluster that you want to manage and click the name or click **Details** in the **Actions** column.
>
> iii. Choose **Applications > Helm**.
>
> iv. On the **Helm** page, delete the ack-node-problem-detector component.

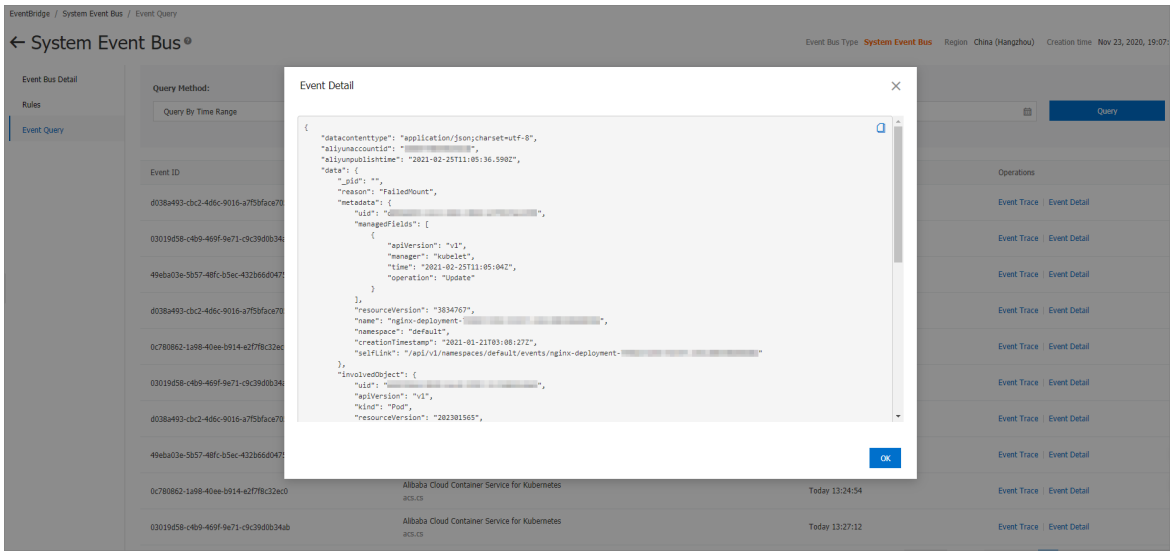
4. On the **App Catalog - ack-node-problem-detector** page, click the **Parameters** tab and modify the following settings:

   Configure the Kubernetes event center and enable EventBridge as a sink of Kubernetes events.

- In the `npd` section, set the `enabled` parameter to true.
- In the `eventer.sinks.eventbridge.enable` section, set the parameter to true.

```
eventbridge:
  # If you want the monitoring results to be notified by EventBridge,
  # first, you need to access to EventBridgeset (https://eventbridge.console.aliyun.com)
  # then set enabled to true.
  # You can visit EventBridge website to get more information. https://eventbridge.console.aliyun.com
  enabled: true
```

5. After the configurations are complete, click **Create** to deploy the ack-node-problem-detector component.
6. After EventBridge is enabled as a sink of Kubernetes events, you can view Kubernetes events in the EventBridge console.
   i. Log on to the EventBridge console.
   ii. In the left-side navigation pane, click **System Event Bus**.
   iii. In the left-side navigation pane of the **System Event Bus** page, click **Event Query**.
   iv. Select a query method and a query range, and click **Search**.
   v. In the list of events, find the event and click **Event Details** in the **Actions** column.



11.3.4. Use Prometheus to monitor a Kubernetes cluster

Prometheus is an open source tool that can be used to monitor cloud-native applications. This topic describes how to deploy Prometheus in a Container Service for Kubernetes (ACK) cluster.

Prerequisites

- A Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群.
- You are connected to the cluster. Make sure that you can view node information such as tags. For more information, see Connect to Kubernetes clusters by using kubectl.

Context

Typically, a monitoring system monitors the following types of object:

- Resources: resource usage of nodes and applications. In a Kubernetes cluster, the monitoring system monitors the resource usage of nodes, pods, and the cluster.

- Applications: internal metrics of applications. For example, the monitoring system dynamically counts the number of online users who are using an application, and enables ports of the application to monitor services and generate alerts.

In a Kubernetes cluster, the monitoring system monitors the following objects:

- Cluster components: the components of the Kubernetes cluster, such as kube-apiserver, kube-controller-manager, and etcd.
- Static resource entities: the node resource status and kernel events.
- Dynamic resource entities: the entities of abstract workloads in Kubernetes, such as Deployments, DaemonSets, and pods.
- Custom objects in applications: the custom data and metrics in applications.

To monitor cluster components and static resource entities, specify the monitoring methods in the configuration files.

To monitor dynamic resource entities for a Kubernetes cluster, you can deploy Prometheus in the Kubernetes cluster.

Procedure

1. Deploy Prometheus.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane, choose **Marketplace > App Catalog**. On the page that appears, click **ack-prometheus-operator**.

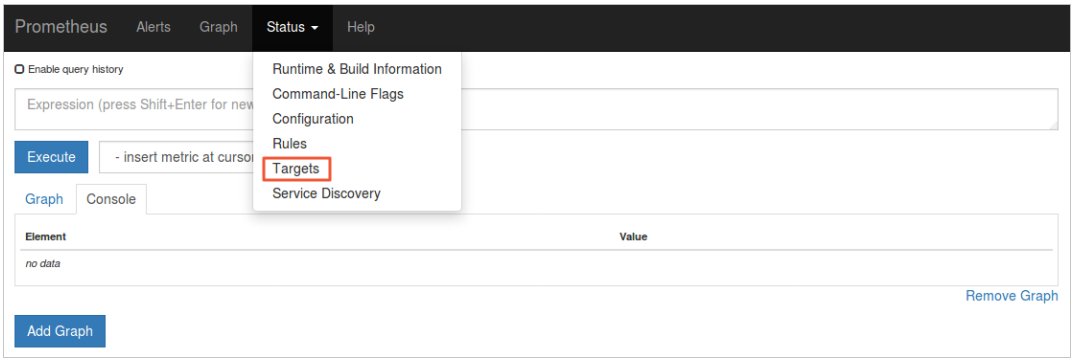
iii. In the **Deploy** section of the **App Catalog - ack-prometheus-operator** page, select the cluster that you want to manage, and click **Create** to deploy Prometheus.
Check the deployment result.

a. Run the following command to map Prometheus in the cluster to local port 9090:

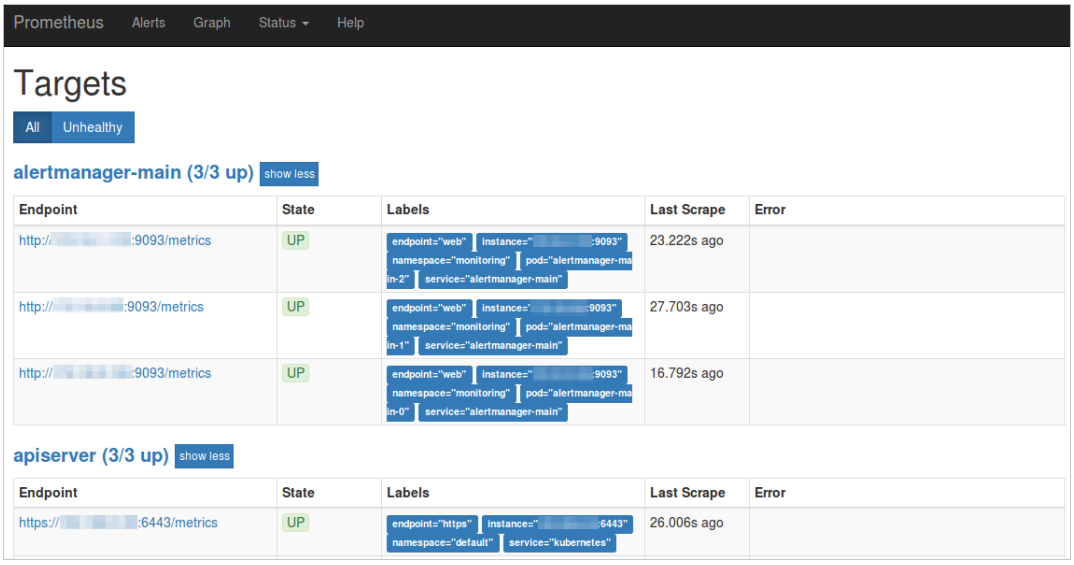
```
kubectl port-forward svc/ack-prometheus-operator-prometheus 9090:9090 -n monitoring
```

b. Enter *localhost:9090* in the address bar of a browser to visit the Prometheus page.

c. In the top navigation bar, choose **Status > Targets** to view all data collection tasks.



Tasks in the **UP** state are running properly.

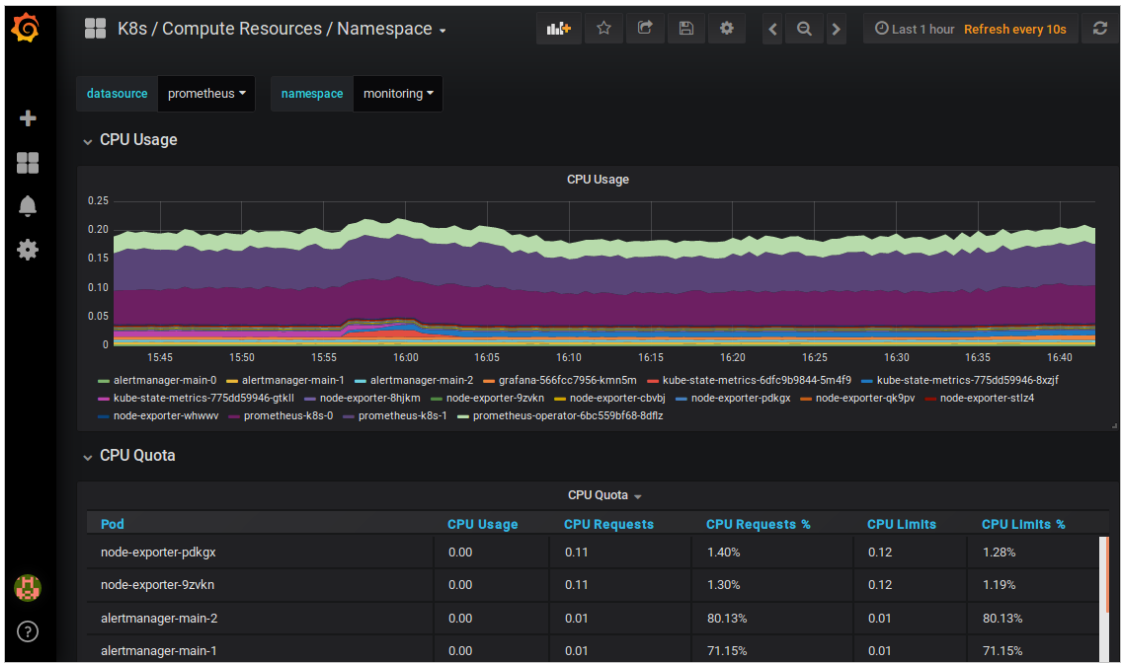


2. View the aggregated data.

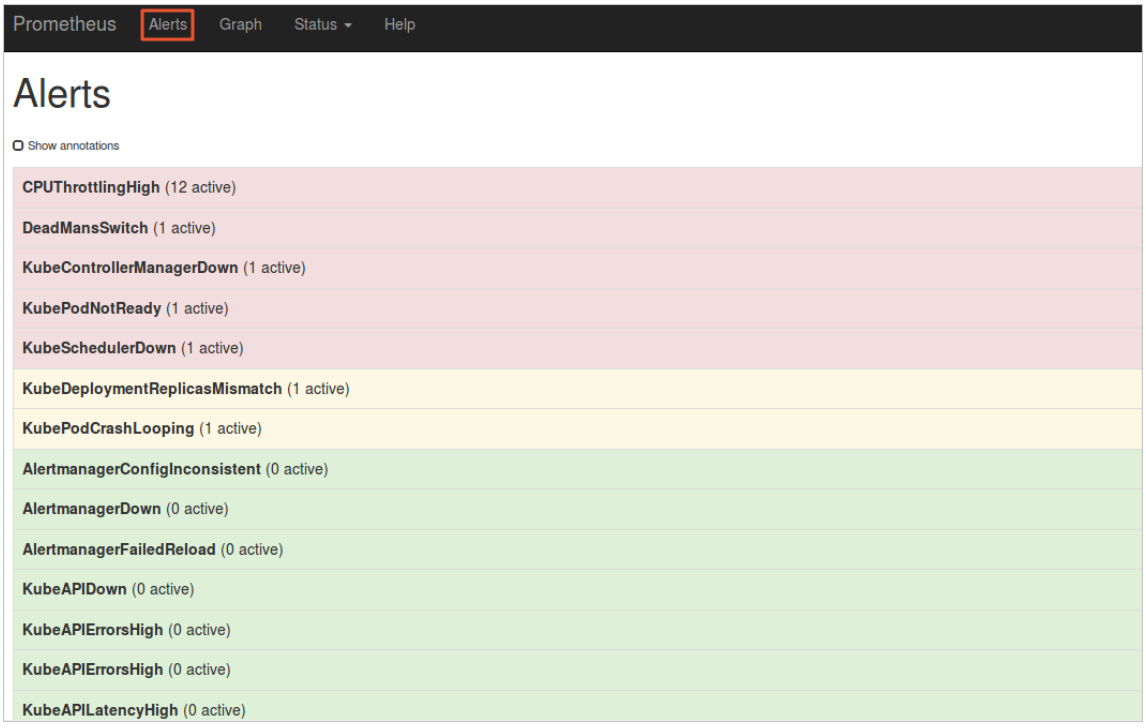
i. Run the following command to map Grafana in the cluster to local port 3000:

```
kubectl -n monitoring port-forward svc/ack-prometheus-operator-grafana 3000:80
```

ii. To view the aggregated data, enter *localhost:3000* in the address bar of a browser, and select a dashboard.



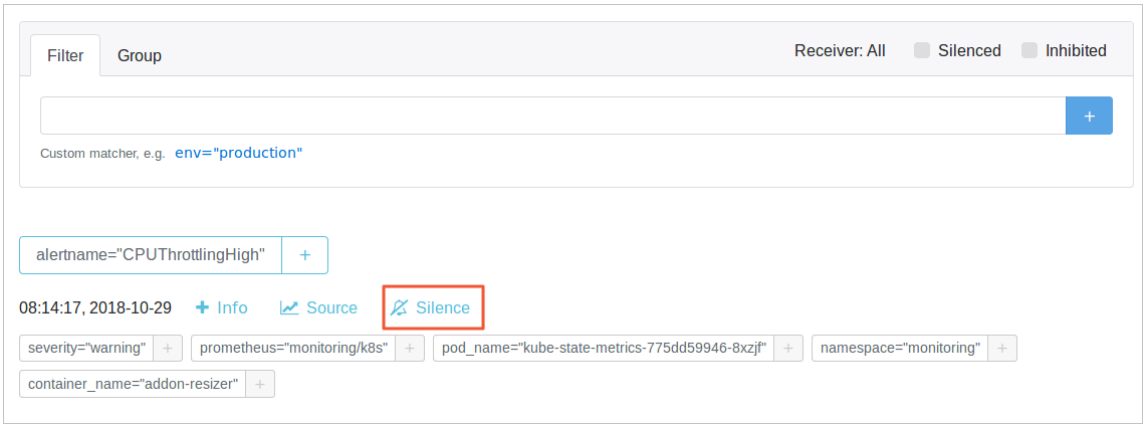
3. View alert rules and set silent alerts.
   - View alert rules
     To view the alert rules, enter `localhost:9090` in the address bar of a browser, and click **Alerts** in the top navigation bar.
     - If an alert rule is in red, alerts are triggered based on this rule.
     - If an alert rule is in green, no alerts are triggered based on this rule.



   - Set silent alerts

Run the following command. Enter `localhost:9093` in the address bar of a browser, and click **Silence** to set silent alerts.

```
kubectl --namespace monitoring port-forward svc/alertmanager-operated 9093
```



You can follow the preceding steps to deploy Prometheus. The following examples describe how to configure Prometheus in different scenarios.

Alert configuration

To set alert notification methods or notification templates, perform the following steps to configure the config field in the alertmanager section:

- Set alert notification methods
  Prometheus that is deployed in a Kubernetes cluster can send notifications by using DingTalk or emails. You can perform the following steps to set alert notification methods:
  - Configure DingTalk notifications
    On the **App Catalog - ack-prometheus-operator** page, click the **Parameters** tab. In the dingtalk field, set enabled to true. In the token field, enter the webhook URLs of your DingTalk chatbots. In the alertmanager section, specify a DingTalk chatbot name for the receiver parameter in the config field. By default, the value of the receiver parameter is webhook.
    For example, if you have two DingTalk chatbots, perform the following steps:

a. Replace the parameter values in the token field with the webhook URLs of the DingTalk chatbots.
Copy the webhook URLs of the DingTalk chatbots and replace the parameter values of dingtalk1 and dingtalk2 in the token field with the copied URLs. In this example, *https://oapi.dingtalk.com/robot/send?access_token=xxxxxxxxxx* is replaced by the webhook URLs.

```
##
podAnnotations: {}
service:
  ## Configuration for dingtalk service
  ##
  annotations: {}
clusterIP: ""
externalIPs: ""
loadBalancerIP: ""
type: ClusterIP
image:
  repository: timonwong/prometheus-webhook-dingtalk
  tag: latest
  pullPolicy: IfNotPresent
# pls fill your dingtalk rebot token
# ref: https://open-doc.dingtalk.com/microapp/serverapi2/qf2nxq
token:
- dingtalk1=https://oapi.dingtalk.com/robot/send?access_token=xxxxxxxxx
- dingtalk2=https://oapi.dingtalk.com/robot/send?access_token=xxxxxxxxx
```

b. Modify the value of the receiver parameter.
In the alertmanager section, find the receiver parameter in the config field, and specify a DingTalk chatbot name for the parameter. In this example, *webhook1* and *webhook2* are used.

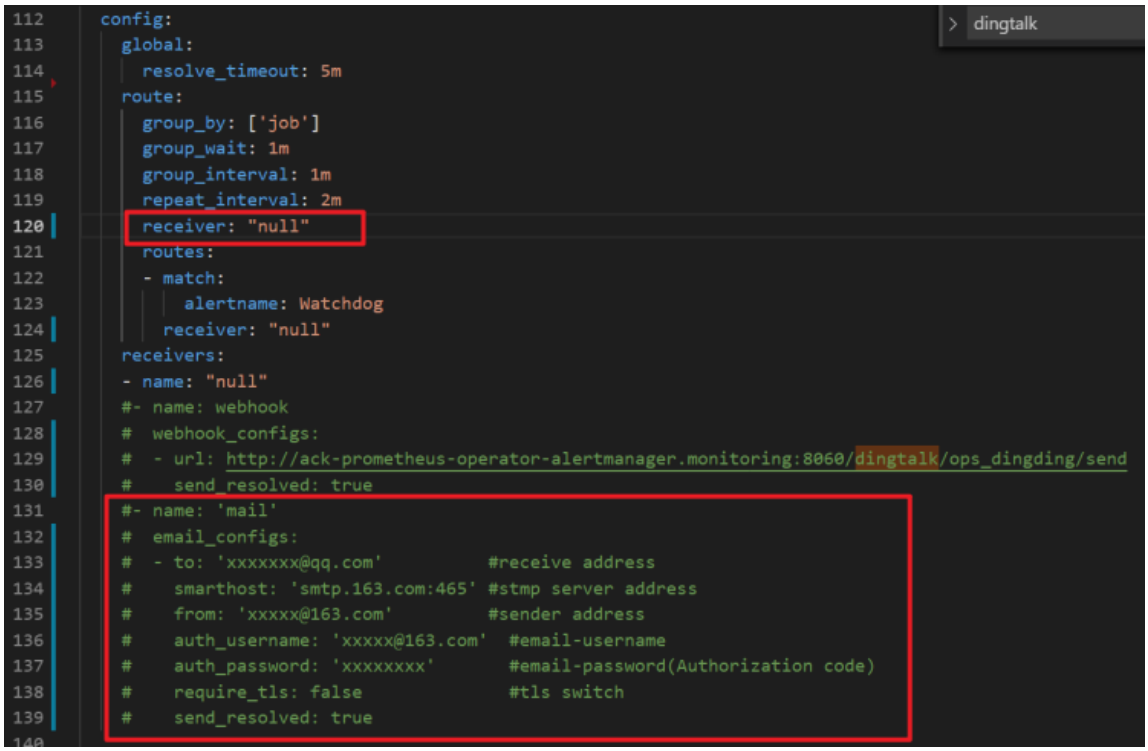
c. Modify the url parameter.
Replace the value of the url parameter with the actual chatbot names. In this example, *dingtalk1* and *dingtalk2* are used.

```
config:
  global:
    resolve_timeout: 5m
  route:
    group_by: ['job']
    group_wait: 1m
    group_interval: 1m
    repeat_interval: 2m
    receiver: "webhook1"
    routes:
    - match:
        alertname: Watchdog
      receiver: "webhook2"
  receivers:
  #- name: "null"
  - name: webhook1
    webhook_configs:
    - url: http://ack-prometheus-operator-dingtalk:8060/dingtalk/dingtalk1/send
      send_resolved: true
  - name: webhook2
    webhook_configs:
    - url: http://ack-prometheus-operator-dingtalk:8060/dingtalk/dingtalk2/send
      send_resolved: true
```

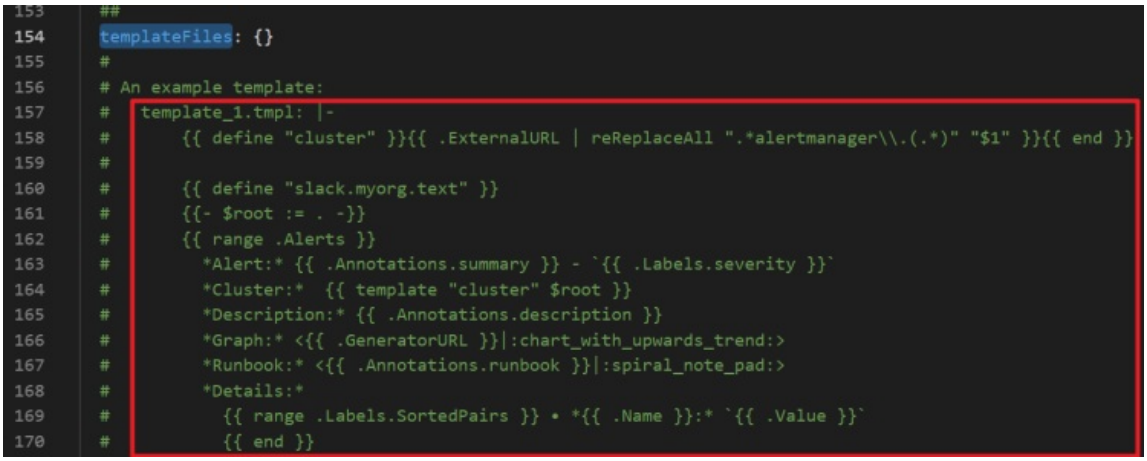
1
2

Note You can add multiple DingTalk chatbots based on your business requirements.

- Configure email notifications
  On the **App Catalog - ack-prometheus-operator** page, click the **Parameters** tab. Set the parameters in the second red box of the following figure. In the alertmanager section, set the receiver parameter in the config field to the email alert name in the receivers field. By default, the email alert name is mail.



- Set alert notification templates
  In the templateFiles field of the alertmanager section on the Parameters tab, you can customize alert notification templates.



Storage configuration

Monitoring data that is generated by Prometheus can be stored in Time Series Database (TSDB) or on disks. You can perform the following steps to configure data storage:

- Store data in TSDB

On the **App Catalog - ack-prometheus-operator** page, click the **Parameters** tab. In the tsdb section, set enabled to true. Then, set the url parameter for remoteRead and remoteWrite.

```
1329      ##  remoteWrite must Indicate.
1330      tsdb:
1331        #default is false
1332        enabled: false
1333        # the tsdb's specification
1334        specification: mlarge
1335        ## The remote_read spec configuration for Prometheus.
1336        ## ref: https://github.com/coreos/prometheus-operator/blo
    .md#remotereadspec
1337        remoteRead:
1338          - url: ""
1339            # read_recent: true
1340        # - url: "http://ts-xxxxxxxxxxxx.hitsdb.rds.aliyuncs.com:
1341        ## The remote_write spec configuration for Prometheus.
1342        ## ref: https://github.com/coreos/prometheus-operator/blo
    .md#remotewritespec
1343        remoteWrite:
1344          - url: ""
1345        # - url: "http://ts-xxxxxxxxxxxx.hitsdb.rds.aliyuncs.com:
```

- Store data on disks
  By default, data collected by Prometheus in Kubernetes is stored on disks. You can configure disk storage in the prometheus or alertmanager section on the Parameters tab. On the **App Catalog - ack-prometheus-operator** page, click the **Parameters** tab. Find the storage field in the alertmanager section or the storageSpec field in the prometheus section, and set the parameters. storageClassName specifies the disk category, accessModes specifies the access mode, and storage specifies the storage capacity.

```
# volumeClaimTemplate:
#   spec:
#     storageClassName: gluster
#     accessModes: ["ReadWriteOnce"]
#     resources:
#       requests:
#         storage: 50Gi
#   selector: {}
```

Note Assume that you want to configure an SSD to store data. In the storageSpec field, set `storageClassName` to *alicloud-disk-ssd*, `accessModes` to *ReadWriteOnce*, and `storage` to *50Gi*, as shown in the following figure.

```
storageSpec:
#  volumeClaimTemplate:
#    spec:
#      storageClassName: alicloud-disk-ssd
#      accessModes: ["ReadWriteOnce"]
#      resources:
#        requests:
#          storage: 50Gi
#    selector: {}
```

To check the configuration, you can choose **Storage & Snapshots > Disks** in the Elastic Compute Service (ECS) console.
For information about how to reuse a disk, see Usage notes for disk volumes.

Enable the Prometheus adapter for auto scaling

To enable the Prometheus adapter, you can set enabled to true in the prometheusAdapter section and customize metrics. In this case, the Kubernetes cluster can automatically scale the number of pods based on the custom metrics. This improves resource usage.

To enable the Prometheus adapter, click the **Parameters** tab on the **App Catalog - ack-prometheus-operator** page. In the prometheusAdapter section, set enabled to true.

```
1460  prometheusAdpater:
1461    affinity:
1462    # the switch for prometheusAdpater
1463    enabled: false
1464    #
1465    replicas: 1
1466    image:
1467      repository: quay.io/coreos/k8s-prometheus-adapter-amd64
1468      tag: v0.4.1
1469      pullPolicy: IfNotPresent
1470    # Specifies whether a service account should be created
1471    serviceAccount:
1472      create: true
```

You can run the following command to verify the configuration. For information about how to customize metrics, see Prometheus adapter.

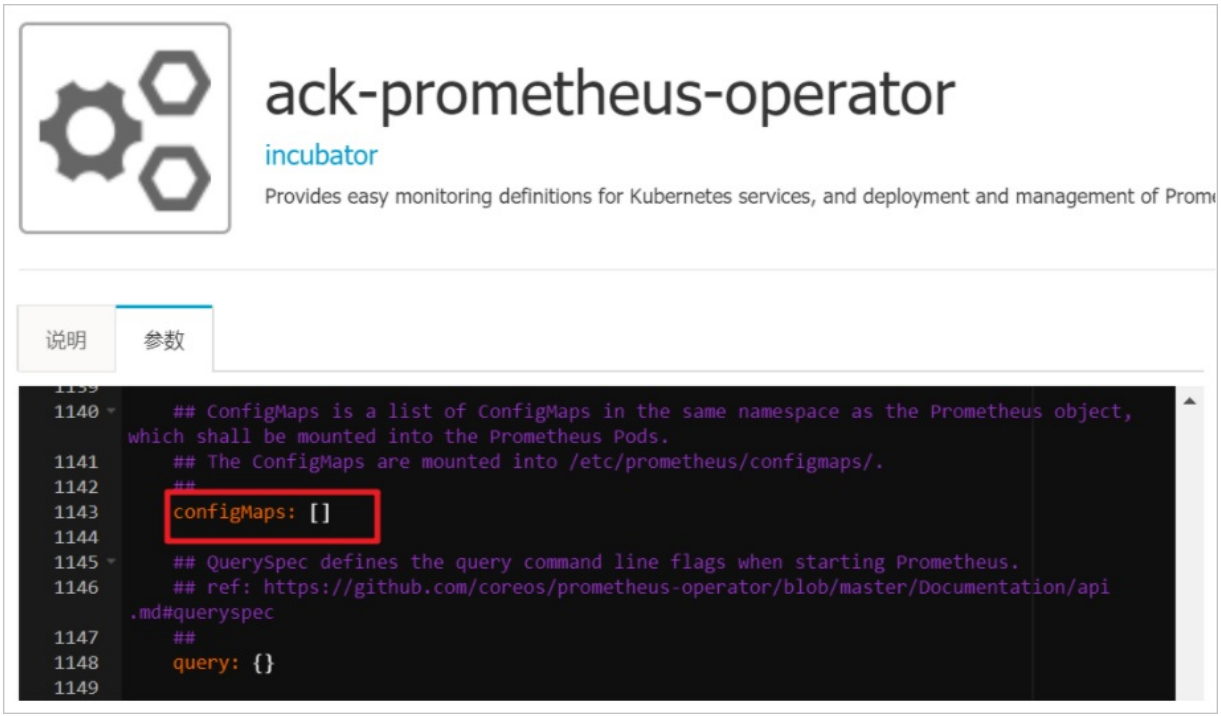
```
kubectl get --raw "/apis/custom.metrics.k8s.io/v1beta1"
```

Mount a custom ConfigMap to Prometheus

This topic describes how to mount a ConfigMap to the */etc/prometheus/configmaps/* path of a pod.

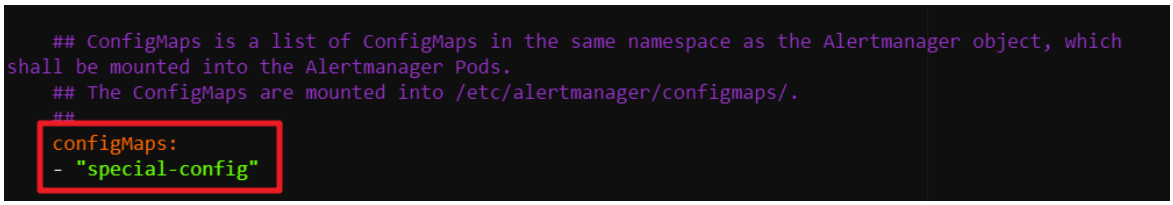
If this is the first time you deploy Prometheus, perform the following steps:

On the **App Catalog - ack-prometheus-operator** page, click the **Parameters** tab. In the configMaps field of the prometheus section, enter the name of the custom ConfigMap.



If Prometheus has been deployed to the cluster, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, click **Releases**.
5. On the **Helm** tab, find the application that you want to update and click **Update** in the Actions column. The **Update Release** dialog box appears.
6. In the configMaps field of the prometheus and alertmanager sections, enter the name of the custom ConfigMap and click **Update**.



Note Assume that you want to mount a custom ConfigMap named `special-config` and this ConfigMap contains the configuration file of Prometheus. To use the configuration file as the value of the --config.file parameter when the Prometheus pod starts, enter special-config in the configMaps field of the prometheus section. After you perform this operation, the ConfigMap is mounted to the Prometheus pod in the */etc/prometheus/configmaps/* path.
The following figure shows an example of the YAML file for special-config.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
data:
  config.yaml: |-
    global:

      [ scrape_interval: <duration> | default = 1m ]


      [ scrape_timeout: <duration> | default = 10s ]


      [ evaluation_interval: <duration> | default = 1m ]


    scrape_configs:
      [ - <scrape_config> ... ]


    rule_files:
      [ - <filepath_glob> ... ]


    alerting:
      alert_relabel_configs:
        [ - <relabel_config> ... ]
      alertmanagers:
        [ - <alertmanager_config> ... ]
```

The following figure shows how to set the configMaps field in the prometheus section.

```
## ConfigMaps is a list of ConfigMaps in the same namespace as the Prometheu
## The ConfigMaps are mounted into /etc/prometheus/configmaps/.
##
configMaps:
- "special-config"
- "detail-config"
```

Grafana configuration

- Mount a dashboard to Grafana
  To mount a dashboard as a ConfigMap to a Grafana pod, click the **Parameters** tab on the **App Catalog - ack-prometheus-operator** page. Set the parameters in the `extraConfigmapMounts` field, as shown in the following figure.

```
extraConfigmapMounts: []
# - name: certs-configmap
#   mountPath: /etc/grafana/ssl/
#   configMap: certs-configmap
#   readOnly: true
```

  > **Note**
  > - Make sure that a dashboard is stored as a ConfigMap in the Kubernetes cluster. This ConfigMap must have a label that is also attached to other ConfigMaps.
  > - In the `extraConfigmapMounts` field of the grafana section, set the ConfigMap name and mount path.
  > - Set mountPath to */tmp/dashboards/*.
  > - Set configMap to the name of your custom ConfigMap.
  > - Set name to the name of the JSON file that stores the dashboard information.

- Enable persistence of dashboards
  You can perform the following steps to enable persistence of Grafana dashboards:
  i. Log on to the ACK console.
  ii. In the left-side navigation pane, click **Clusters**.
  iii. On the **Clusters** page, click the name of the cluster that you want to manage or click **Applications** in the **Actions** column.
  iv. In the left-side navigation pane, click **Releases**.
  v. Find ack-prometheus-operator and click **Update** in the Actions column. The **Update Release** dialog box appears.
  vi. In the Update Release dialog box, find the persistence field in the grafana section and set the parameters, as shown in the following figure.

```
428   persistence:
429     #Persistent switch
430     enabled: false
431     #storageClass Name,there are four types of storageclasses: alicloud-disk-available,alicloud-disk
    -efficiency,alicloud-disk-essd,alicloud-disk-ssd
432     storageClassName: alicloud-disk-efficiency
433     accessModes:
434       - ReadWriteOnce
435     size: 30Gi
436     # annotations: {}
437     # subPath: ""
438     #existingClaim:
```

  You can export Grafana dashboards in JSON format. For more information, see Export a Grafana dashboard.

FAQ

- What can I do if I fail to receive DingTalk alert notifications?
  i. Obtain the webhook URL of your DingTalk chatbot. For more information, see Scenario 3: Use DingTalk

to generate alerts upon Kubernetes events.

ii. On the Parameters tab, find the dingtalk section, set enabled to true, and then enter the webhook URL of your DingTalk chatbot in the token field. For more information, see **Configure DingTalk alert notifications** in Alert configuration.

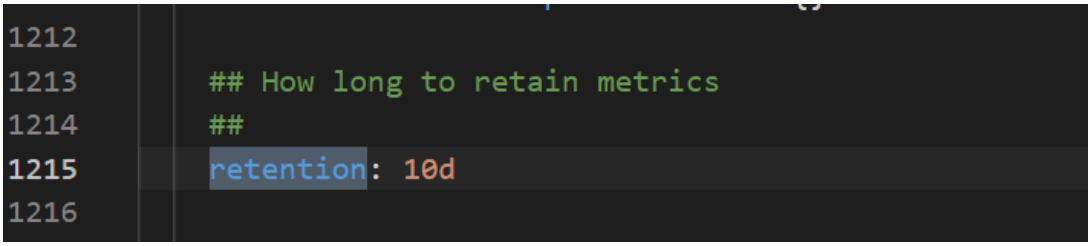
- What can I do if the following error message appears when I deploy Prometheus in a cluster?
  The error message is:

  ```
  Can't install release with errors: rpc error: code = Unknown desc = object is being deleted: customresourcedefinitions.apiextensions.k8s.io "xxxxxxxx.monitoring.coreos.com" already exists
  ```

  The error message indicates that the cluster fails to clear custom resource definitions (CRDs) of the previous deployment. Run the following commands to delete CRDs and then deploy Prometheus again:

  ```
  kubectl delete crd prometheuses.monitoring.coreos.com
  kubectl delete crd prometheusrules.monitoring.coreos.com
  kubectl delete crd servicemonitors.monitoring.coreos.com
  kubectl delete crd alertmanagers.monitoring.coreos.com
  ```

- What can I do if I fail to receive email alert notifications?
  Make sure that the value of `smtp_auth_password` is the SMTP authorization code instead of the logon password of the email account. Make sure that the STMP server endpoint includes a port number.
- What can I do if the following error message appears when I update a YAML file?
  If the configuration file of Tiller is overlarge, the cluster cannot be accessed. To solve this issue, you can delete some annotations in the configuration file and mount the file to a pod as a ConfigMap. You can enter the name of the custom ConfigMap in the configMaps field of the prometheus and alertmanager sections. For more information, see the second method in Mount a custom ConfigMap to Prometheus.
- How do I enable the features of Prometheus after I deploy it in a cluster?
  On the management page of the cluster where Prometheus is deployed, click **Releases** in the left-side navigation pane, and click the **Helm** tab. Find **ack-prometheus-operator** and click **Update** in the Actions column. In the Update Release dialog box, set enabled to true for the features that you want to enable and click **Update**.
- How do I select data storage: TSDB or disks?
  TSDB storage is available to limited regions. However, disk storage is supported in all regions. The following figure shows the data retention policy.

- What can I do if a Grafana dashboard fails to display data?
  On the management page of the cluster, click **Releases** in the left-side navigation pane, and click the **Helm** tab. Find **ack-prometheus-operator** and click **Update** in the Actions column. In the Update Release dialog box, find **clusterVersion** and check whether the value of clusterVersion is correct. If the cluster version is earlier than V1.16, set clusterVersion to 1.14.8-aliyun.1. If the cluster version is V1.16 or later, set clusterVersion to 1.16.6-aliyun.1.
- What can I do if I fail to install Prometheus after I delete the namespace of Prometheus?
  After you delete the namespace of Prometheus, the settings of the resources may be retained. In this case, you may fail to install Prometheus again. You can perform the following operations to delete resource settings:

  i. Delete role-based access control (RBAC) permissions.

a. Delete the ClusterRole.

```
kubectl delete ClusterRole ack-prometheus-operator-grafana-clusterrole
kubectl delete ClusterRole ack-prometheus-operator-kube-state-metrics
kubectl delete ClusterRole psp-ack-prometheus-operator-kube-state-metrics
kubectl delete ClusterRole psp-ack-prometheus-operator-prometheus-node-exporter
kubectl delete ClusterRole ack-prometheus-operator-operator
kubectl delete ClusterRole ack-prometheus-operator-operator-psp
kubectl delete ClusterRole ack-prometheus-operator-prometheus
kubectl delete ClusterRole ack-prometheus-operator-prometheus-psp
```

b. Delete the ClusterRoleBinding.

```
kubectl delete ClusterRoleBinding ack-prometheus-operator-grafana-clusterrolebinding
kubectl delete ClusterRoleBinding ack-prometheus-operator-kube-state-metrics
kubectl delete ClusterRoleBinding psp-ack-prometheus-operator-kube-state-metrics
kubectl delete ClusterRoleBinding psp-ack-prometheus-operator-prometheus-node-exporter
kubectl delete ClusterRoleBinding ack-prometheus-operator-operator
kubectl delete ClusterRoleBinding ack-prometheus-operator-operator-psp
kubectl delete ClusterRoleBinding ack-prometheus-operator-prometheus
kubectl delete ClusterRoleBinding ack-prometheus-operator-prometheus-psp
```

ii. Delete the CustomResourceDefinition (CRD).

```
kubectl delete crd alertmanagerconfigs.monitoring.coreos.com
kubectl delete crd alertmanagers.monitoring.coreos.com
kubectl delete crd podmonitors.monitoring.coreos.com
kubectl delete crd probes.monitoring.coreos.com
kubectl delete crd prometheuses.monitoring.coreos.com
kubectl delete crd prometheusrules.monitoring.coreos.com
kubectl delete crd servicemonitors.monitoring.coreos.com
kubectl delete crd thanosrulers.monitoring.coreos.com
```

11.3.5. Enable ARMS Prometheus

You can view dashboards and performance metrics preset for Container Service for Kubernetes (ACK) by using Application Real-Time Monitoring Service (ARMS) Prometheus. This topic describes how to enable ARMS Prometheus in ACK, how to configure alerts in ARMS Prometheus, and how to customize monitoring metrics and use Grafana to display monitoring metrics.

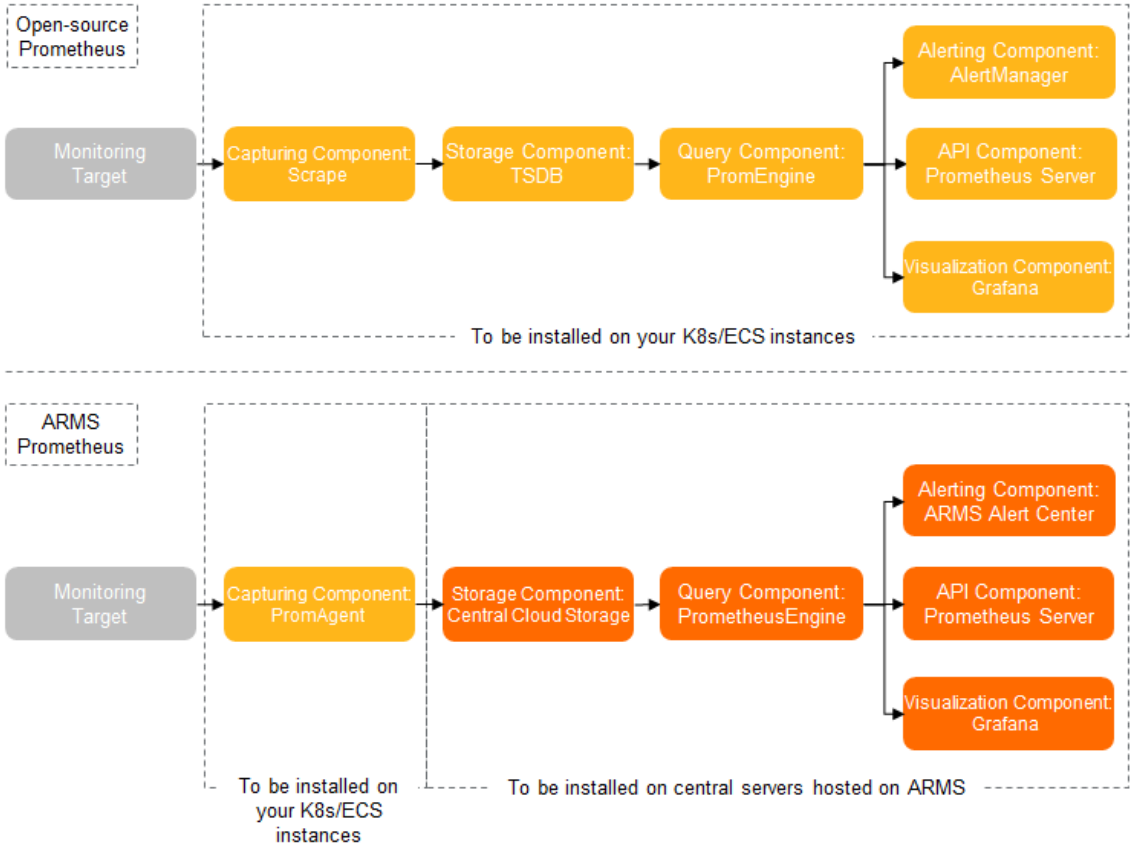
Context

ARMS Prometheus is fully compatible with the open source Prometheus ecosystem. It monitors a wide array of components and provides multiple out-of-the-box dashboards. ARMS Prometheus also provides fully managed monitoring services. ARMS Prometheus saves you the efforts of managing the underlying services, such as data storage, data presentation, and system operations and maintenance (O&M).

Compared with the open source Prometheus, Alibaba Cloud Prometheus Service has the following advantages:

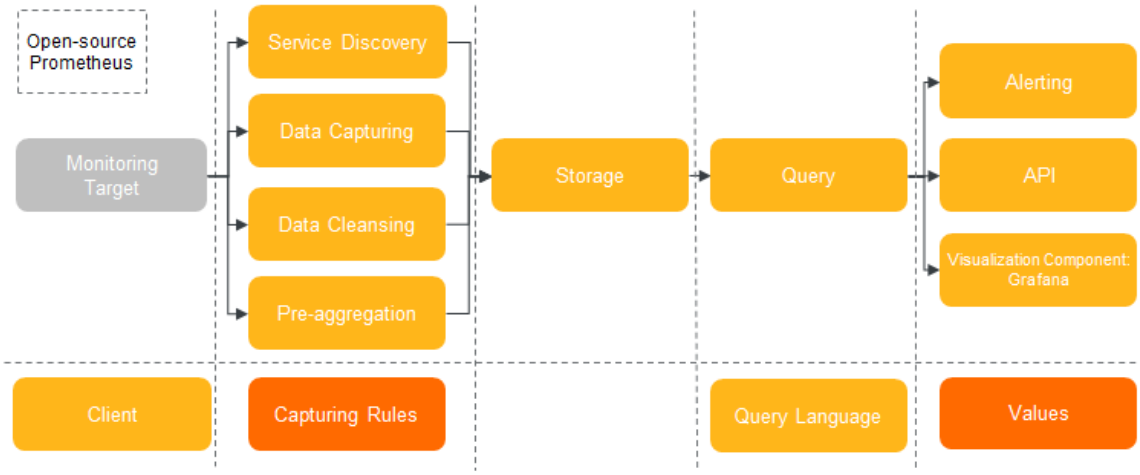
- Lightweight, stable, and accurate retry mechanism

- Compared with the open source Prometheus, the deployment of Alibaba Cloud Prometheus Service is more lightweight. Instead of building a Prometheus monitoring system, you can install the Prometheus agent (PromAgent) to monitor your business.



  - The open source Prometheus consumes 16 GB to 128 GB of memory. Alibaba Cloud Prometheus Service consumes only 200 MB to 1 GB of memory and 1 CPU core. Alibaba Cloud Prometheus Service provides higher system stability than the open source Prometheus.
  - The open source Prometheus retrieves data only once, and data may be discarded when it is written to the storage component. Alibaba Cloud Prometheus Service retries multiple times if it fails to retrieve data. It ensures high concurrency when it writes data to the storage component. No data is discarded.
- Unlimited amount of data
  - The open source Prometheus can collect data based on up to one million metrics. Alibaba Cloud Prometheus Service can scale its data collection capability based on the number of Kubernetes replicas. Collection tasks can be distributed across replicas.
  - The maximum storage capacity of the open source Prometheus is limited by the size of the local disk. Alibaba Cloud Prometheus Service uses the central cloud storage service. The storage capacity is not limited.
- Compatibility with open source systems

- Alibaba Cloud Prometheus Service is compatible with the clients and query languages in the open source Prometheus ecosystem. Alibaba Cloud Prometheus Service provides optimized collection rules and usage values.



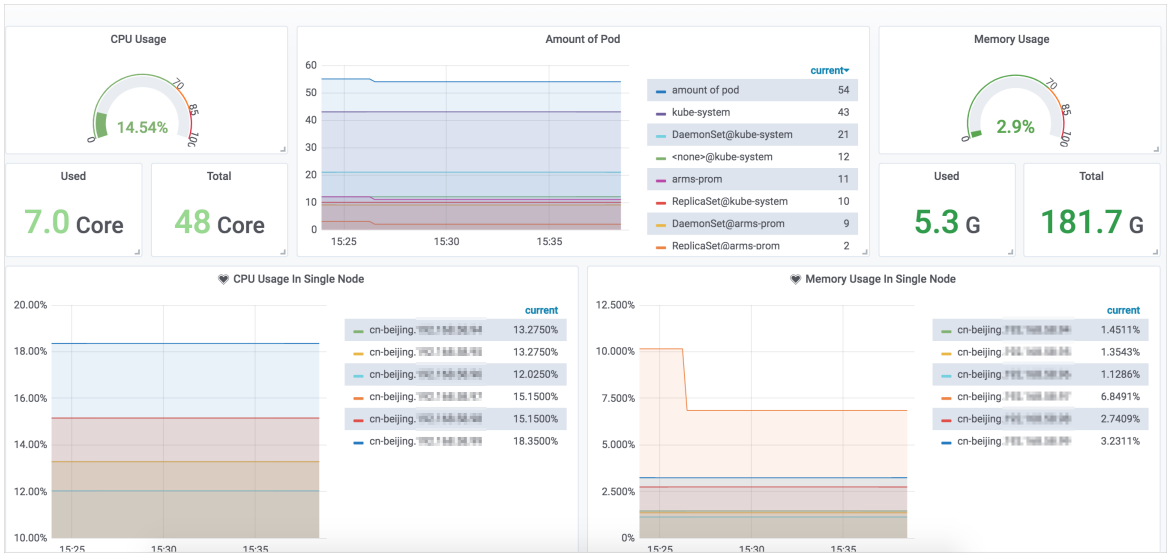
- Alibaba Cloud Prometheus Service is compatible with three mainstream collection rules: the open source *prometheus.yaml* configuration file, ServiceMonitor, and the default collection rule named Annotation. ServiceMonitor is suitable for the monitoring of custom Kubernetes clusters. Compared with the open source Prometheus, Alibaba Cloud Prometheus Service allows you to update collection rules by using the *prometheus.yaml* configuration file. You do not need to write multiple lines of code in the *Deployment* file. You only need to add the following three Annotations.

```
prometheus.io/scrape: "true"
prometheus.io/port: "9090"
prometheus.io/path: "/metrics"
```

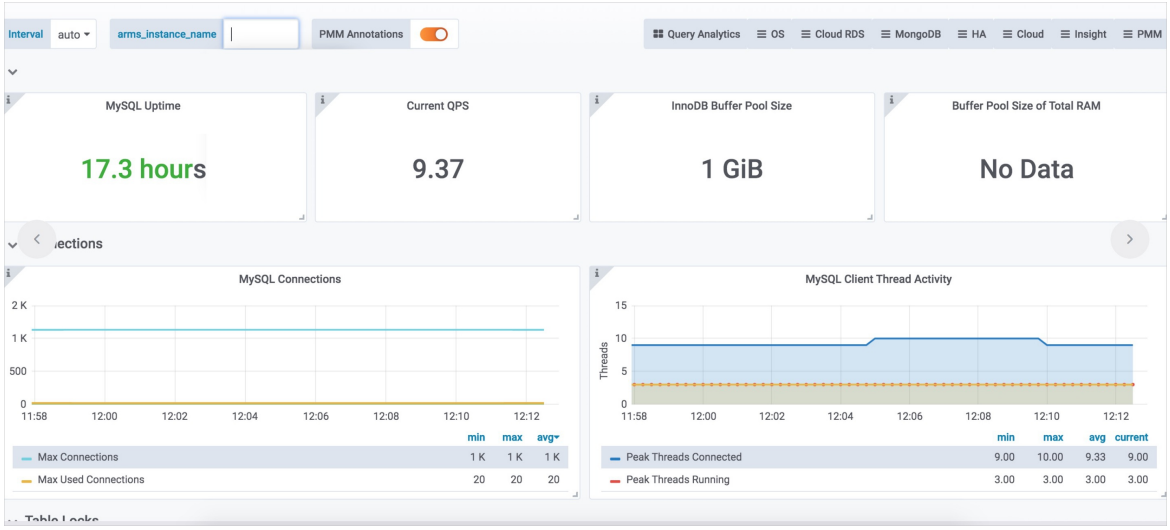
- Alibaba Cloud Prometheus Service allows you to visualize data by using Grafana. By configuring the Prometheus HTTP API URL, you can implement multi-tenant isolation for the data source in Grafana and multi-tenant isolation for the Grafana dashboard. Alibaba Cloud Prometheus Service is also compatible with the Explore data debugging module of Grafana.
- Alibaba Cloud Prometheus Service is compatible with the HTTP API module of the open source Prometheus. It supports three standard API operations: query, query_range, and labelValues. In addition, you can add */userId/clusterId/regionId/* to the data URL to achieve multi-tenant isolation.
- Alibaba Cloud Prometheus Service uses the built-in alerting system of Application Real-Time Monitoring Service (ARMS) and is compatible with the alert rules of the open source Prometheus.

- Cost-effectiveness

- Alibaba Cloud Prometheus Service supports the default Kubernetes monitoring. After you install the default Kubernetes monitoring, Alibaba Cloud Prometheus Service automatically creates exporters, collection rules, Grafana dashboards, and ARMS alerts. Compared with the open source Prometheus, Alibaba Cloud Prometheus Service reduces your time cost from about 3 days to about 10 minutes.



- Alibaba Cloud Prometheus Service supports the monitoring of open source components. You only need to enter the AccessKey ID and AccessKey secret of your Alibaba Cloud account, and the accounts and passwords of the RDS and Redis components. Alibaba Cloud Prometheus Service creates exporters and dashboards for these components. Compared with the open source Prometheus, Alibaba Cloud Prometheus Service reduces your time cost from about 7 days to about 3 minutes.



- Alibaba Cloud Prometheus Service supports easy installation and removal. You can debug the service by performing health checks. Compared with the open source Prometheus, Alibaba Cloud Prometheus Service reduces your time cost from about 1 day to about 3 minutes.

For more information about ARMS Prometheus, see What is Prometheus Service?.

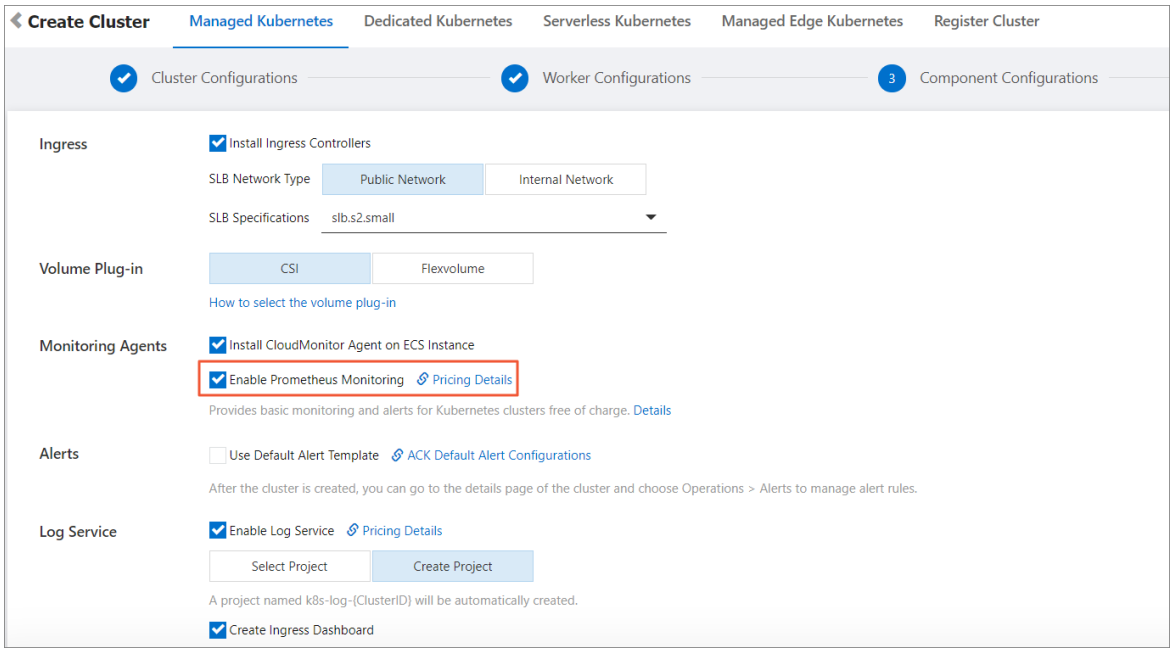
Enable ARMS Prometheus

You can use one of the following methods to enable ARMS Prometheus:

Enable ARMS Prometheus by setting cluster parameters

1. Log on to the ACK console.

2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Create Kubernetes Cluster**.
4. Select the cluster template that you want to use and set the cluster parameters. On the **Component Configurations** wizard page, select **Enable Prometheus Monitoring**.



For more information about how to create a cluster, see 创建Kubernetes托管版集群.

> **Note** By default, **Enable Prometheus Monitoring** is selected when you create a cluster.

After the cluster is created, the system automatically configures ARMS Prometheus.

Enable ARMS Prometheus on the Prometheus Monitoring page in the ACK console

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the cluster details page, choose **Operations > Prometheus Monitoring**.
5. In the middle of the **Prometheus Monitoring** page, click **Install**.

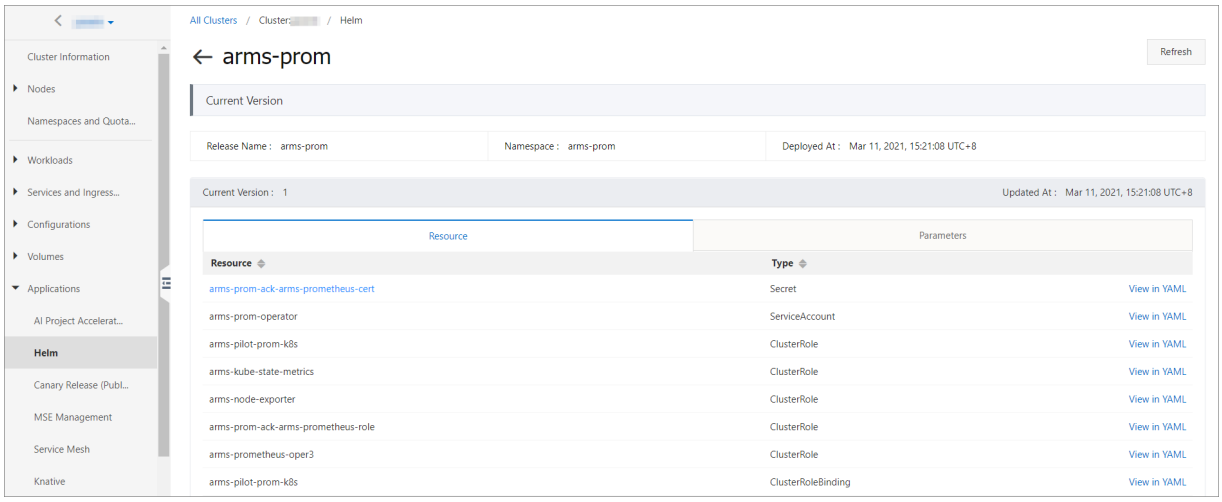
Enable ARMS Prometheus on the App Catalog page in the ACK console

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab. Then, find and click **ack-arms-prometheus**.
4. On the **App Catalog - ack-arms-prometheus** page, select the cluster for which you want to enable ARMS Prometheus in the **Deploy** section, and click **Create**.

> **Note** By default, **Namespace** and **Release Name** are set to **arms-prom**.

Execution result

After the installation is completed, the **arms-prom** page appears. You can view application information on this page.



View Grafana dashboards provided by ARMS Prometheus

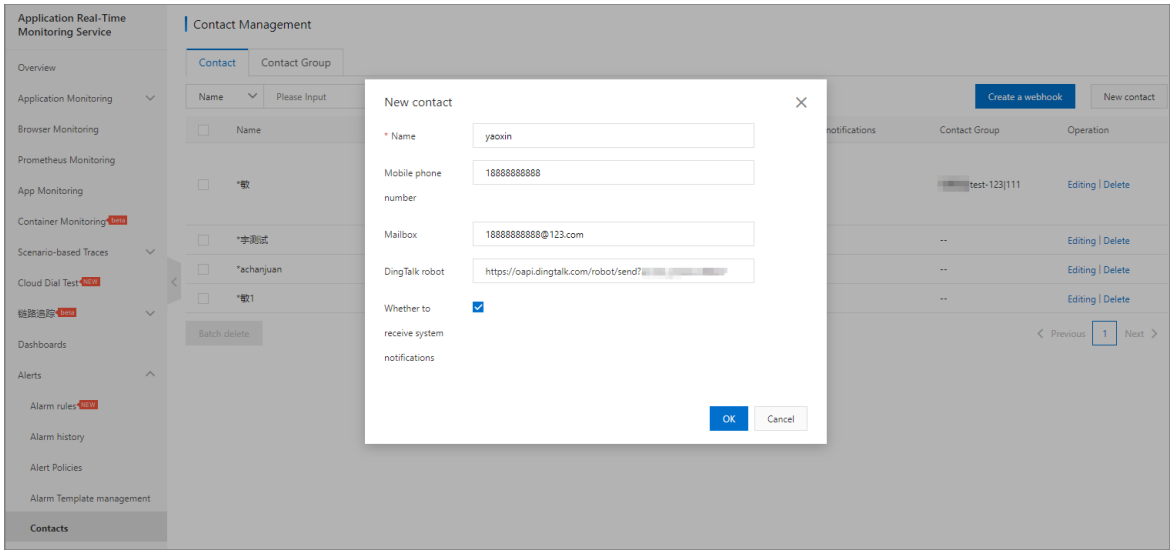
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Prometheus Monitoring**.
5. On the **Prometheus Monitoring** page, click the name of a Grafana dashboard to view the monitoring data.

Configure alerts in ARMS Prometheus

ARMS Prometheus allows you to create alerts for monitoring jobs. When alert conditions are met, you can receive alerts through emails, Short Message Service (SMS) messages, and DingTalk notifications in real time. This helps you detect errors in a proactive manner. When an alert rule is triggered, notifications are sent to the specified contact group. Before you can create a contact group, you must create a contact. When you create a contact, you can specify a mobile number and an email address which the contact can use to receive notifications. You can also provide the webhook URL of a DingTalk chatbot that can automatically send alert notifications.

> **Note** To add a DingTalk chatbot as a contact, you must first obtain the webhook URL of the chatbot. For more information, see Configure a DingTalk chatbot to send alert notifications.

1. Log on to the ARMS console .
2. In the left-side navigation pane, choose **Alerts > Contacts**.
3. In the upper-right corner of the **Contact** tab, click **New contact** Configure the contact and click **OK**.



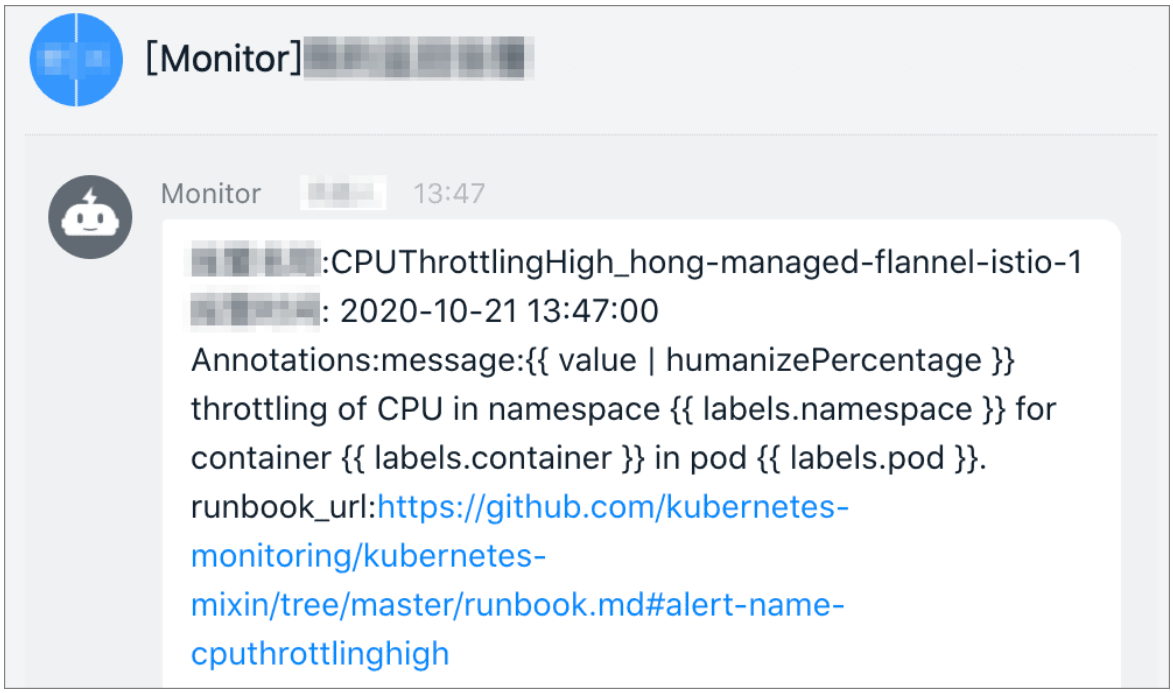
4. Configure an alert rule.
   i. Log on to the ARMS console.
   ii. In the left-side navigation pane, click **Prometheus Monitoring**.
   iii. On the **Prometheus Monitoring** page, click the name of a Kubernetes cluster in the **K8s** column.
   iv. In the left-side navigation pane, click **Alarm configuration**.
   v. Select the alert that you want to manage and click **Editing** in the **Actions** column. Modify the PromQL statement and click **OK**. You can also select a preset alert and click **Enable** to enable the alert.

      For more information about how to configure PromQL statements, see Create ARMS alerts.

> **Note** You can also choose **Alerts > Alert Policies** in the ARMS console to manage alerts.

Verify the result

Perform a manual test to trigger an alert in DingTalk. The following figure shows a sample alert.

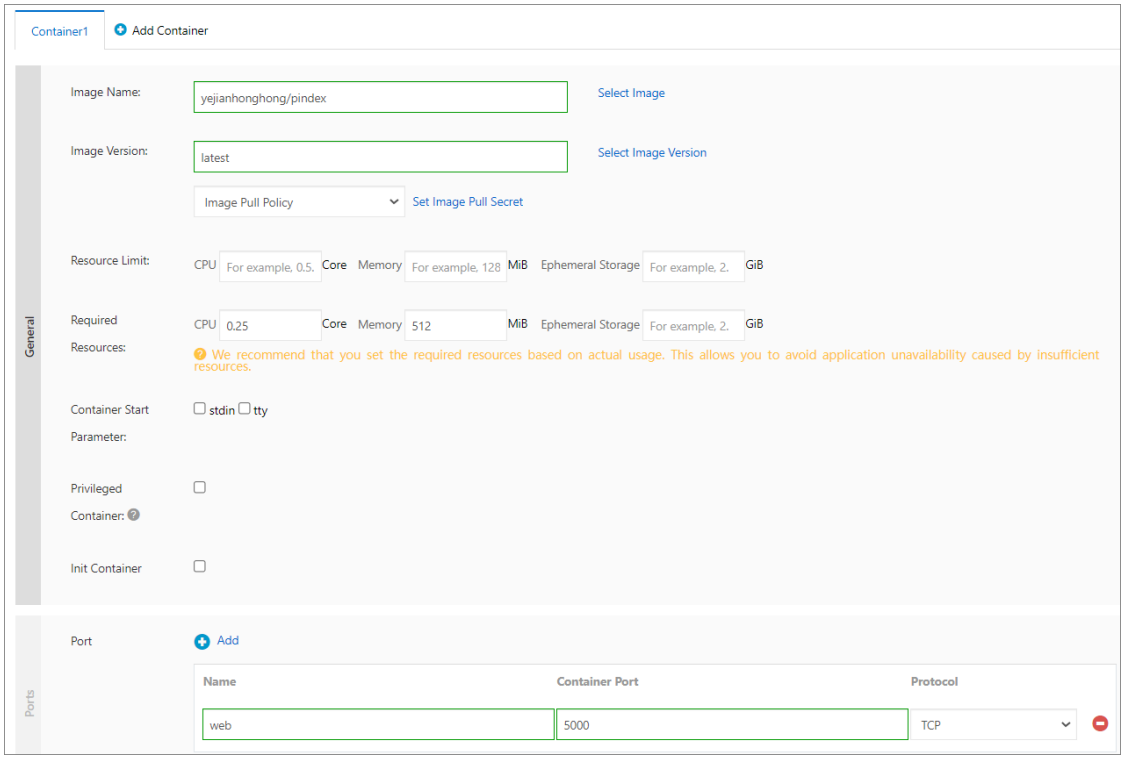


Customize monitoring metrics and use Grafana to display monitoring metrics

Method 1: Use annotations to customize monitoring metrics

You can add annotations to pod configuration templates to define custom monitoring metrics. The application monitoring component of ARMS uses ARMS Prometheus to automatically obtain these custom monitoring metrics.

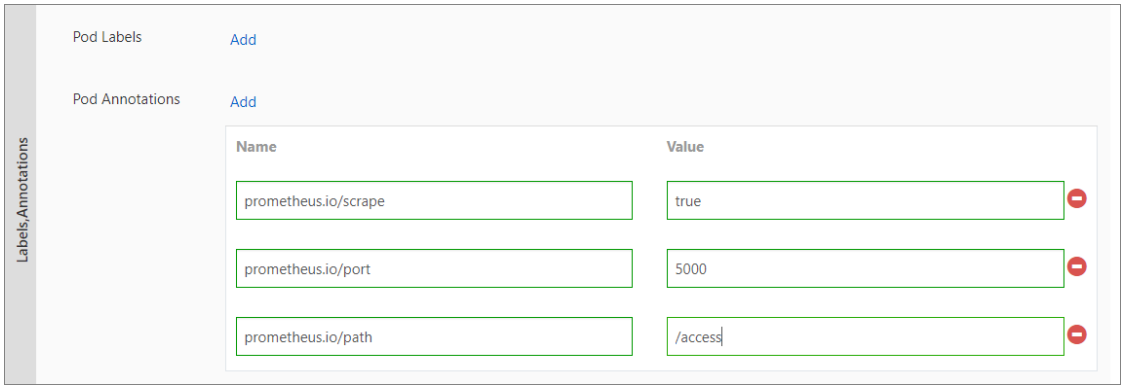
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, create an application.
   i. Click **Create from Image**.
   ii. On the **Basic Information** wizard page, set basic parameters and click **Next**.
   iii. Create a web application and open port 5000 for the application.
   In this example, the `yejianhonghong/pindex` image is used.



   iv. Click **Next**.

v. Add annotations that are related to ARMS to the pod.

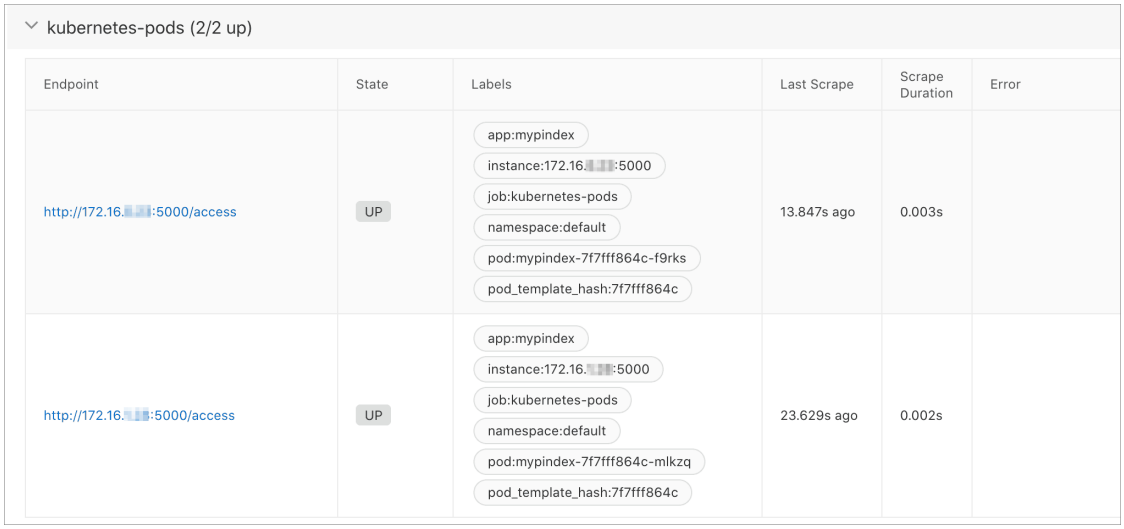
The *prometheus.io/port* annotation is used to specify the endpoint port that ARMS Prometheus scrapes. The *prometheus.io/path* annotation is used to specify the endpoint path that ARMS Prometheus scrapes.



vi. Click **Create** to create the application.

6. On the **Services** page, create a Service.
   i. In the left-side navigation pane, choose **Services and Ingresses > Services**.
   ii. In the upper-right corner of the **Services** page, click **Create**.
   iii. Select **Server Load Balancer** and **Public Access** for **Type**.
   iv. Select the application that is created in Step 4 for **Backend**.
   v. Click **Create** to create the Service.

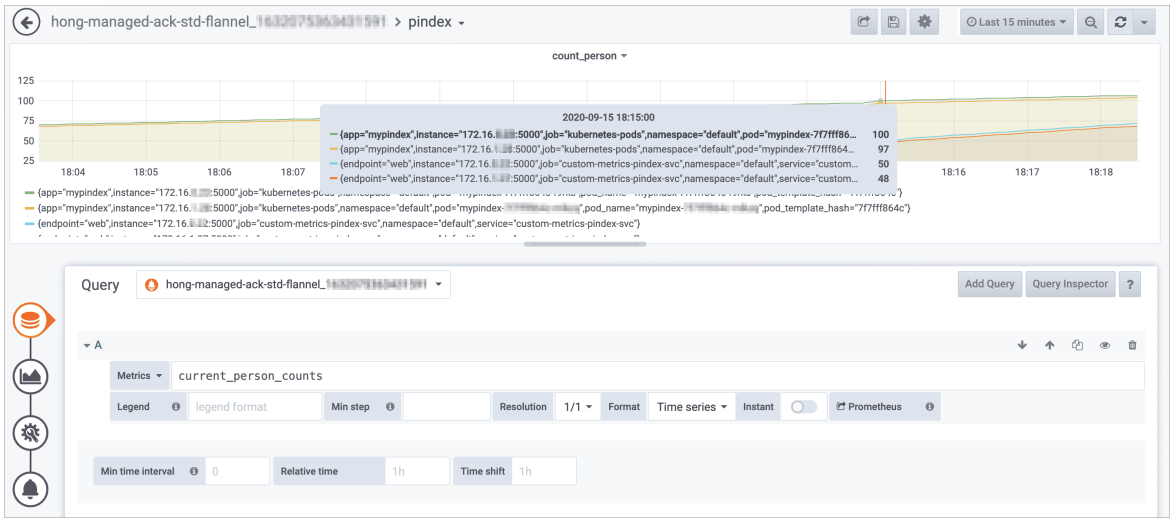
   For more information, see Manage Services.
7. Configure custom monitoring metrics.
   i. Log on to the ARMS console.
   ii. In the left-side navigation pane, click **Prometheus Monitoring**.
   iii. On the **Prometheus Monitoring** page, click the name of a Kubernetes cluster in the **K8s** column.
   iv. In the left-side navigation pane, click **Settings**. Click the **Targets (beta)** tab and verify that the custom metrics are configured.



8. Access the public IP address of the Service that is created in Step 5. This increases the value of the following custom metric.

For more information about metrics, see Data model.

9. Go to the Dashboards page of the ARMS console and click a dashboard to go to the Grafana page. Click **Add panel** in the upper-right corner, select the **Graph** type, and then enter *current_person_counts* for **Metrics**.
10. Save the settings to view the Grafana chart of the custom metric.



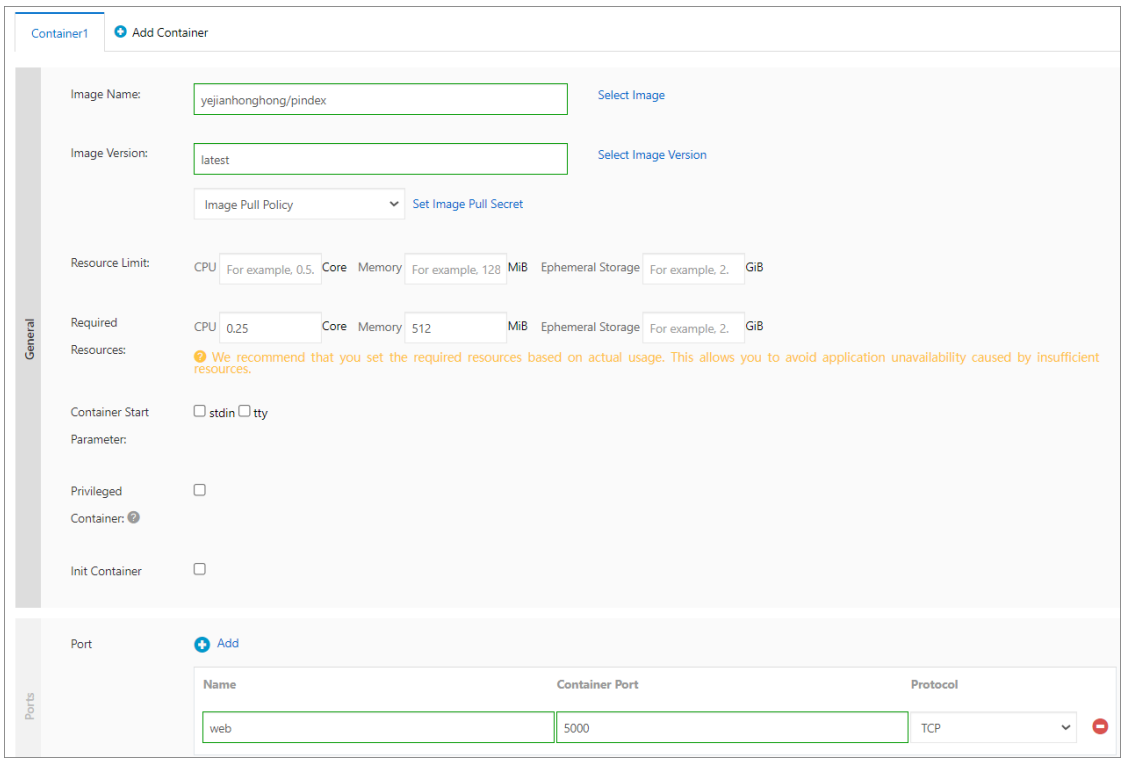
Method 2: Use ServiceMonitors to customize monitoring metrics

To use ServiceMonitors to customize monitoring metrics, you must add labels to Services. You do not need to add annotations.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the **Deployments** page, create an application.
   i. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
   ii. Click **Create from Image**.
   iii. On the **Basic Information** wizard page, set basic parameters and click **Next**.

iv. Create a web application and open port 5000 for the application.

In this example, the `yejianhonghong/pindex` image is used.



Click **Next**.

v. Click **Create** to create the application.

5. On the **Services** page, create a Service.

i.

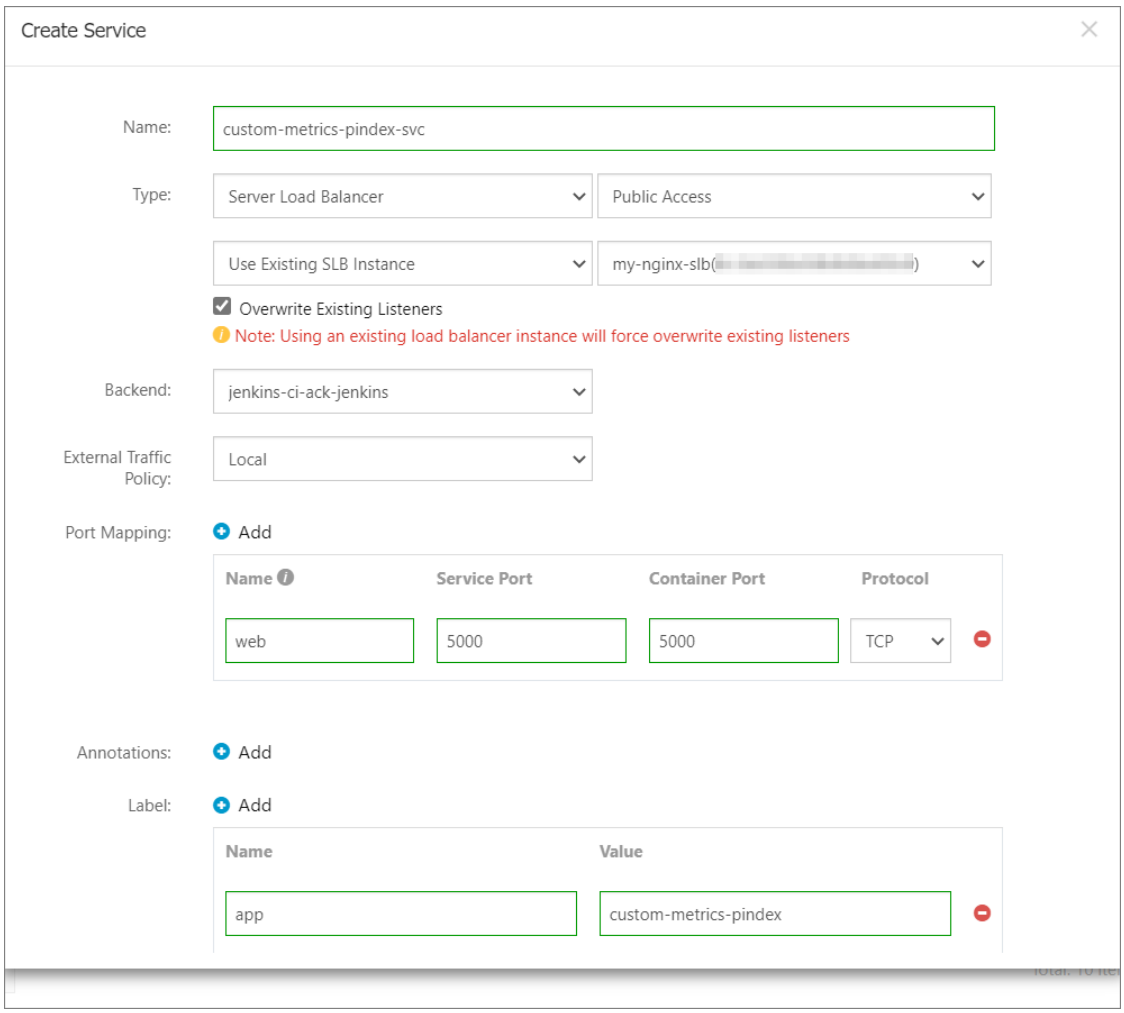
ii. In the upper-right corner of the **Services** page, click **Create**.

iii. Select **Server Load Balancer** and **Public Access** for **Type**.

iv. Select the application that is created in Step 4 for **Backend**.

v. Add the following label to the Service.

This label is used by ServiceMonitor as a selector.



vi. Click **Create** to create the Service.

For more information, see Manage Services.

6. Specify the endpoint that ARMS Prometheus scrapes.
   i. Log on to the Prometheus console .
   ii. In the left-side navigation pane, click **Prometheus Monitoring**. Click the cluster that you want to manage in the **K8s** column.
   iii. In the left-side navigation pane, click **Settings**. Then, click the **Service Discovery** tab.

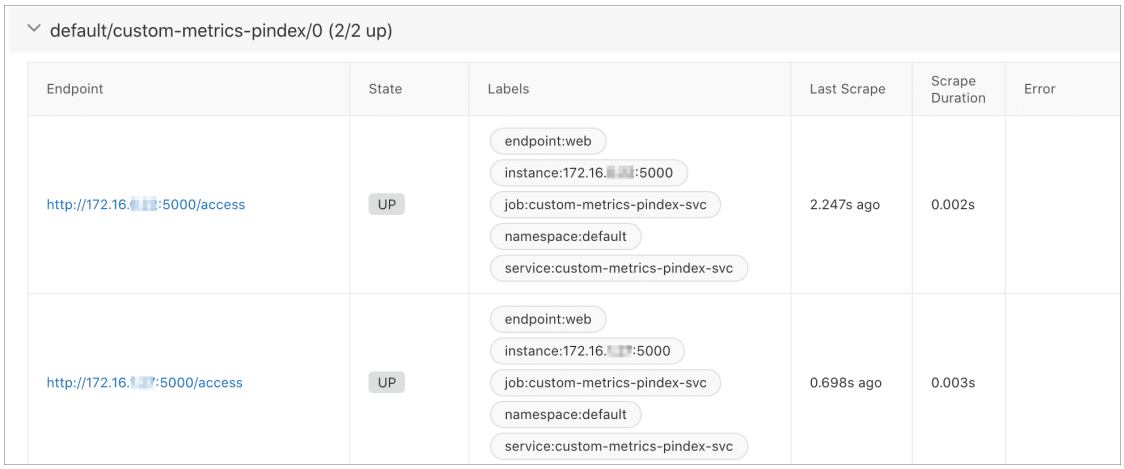
iv. Click the **ServiceMonitor** tab. In the upper-right corner of the tab, click **Add ServiceMonitor**.

In this example, the following template is used to create a ServiceMonitor.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
 # Enter a unique name.
 name: custom-metrics-pindex
 # Specify a namespace.
 namespace: default
spec:
 endpoints:
 - interval: 30s
  # Enter the name of the port specified in the Port Mapping section when you created the Service in Ste
p 5.
  port: web
  # Enter the path of the Service.
  path: /access
 namespaceSelector:
  any: true
  # The namespace of the NGINX demo application.
 selector:
  matchLabels:
   # Enter the label that you added to the Service in Step 5.
   app: custom-metrics-pindex
```

Click **OK** to create the ServiceMonitor.

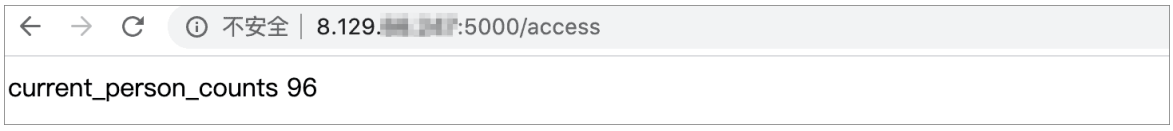
v. On the **Targets (beta)** tab, verify that the endpoints that ARMS Prometheus scrapes are displayed.



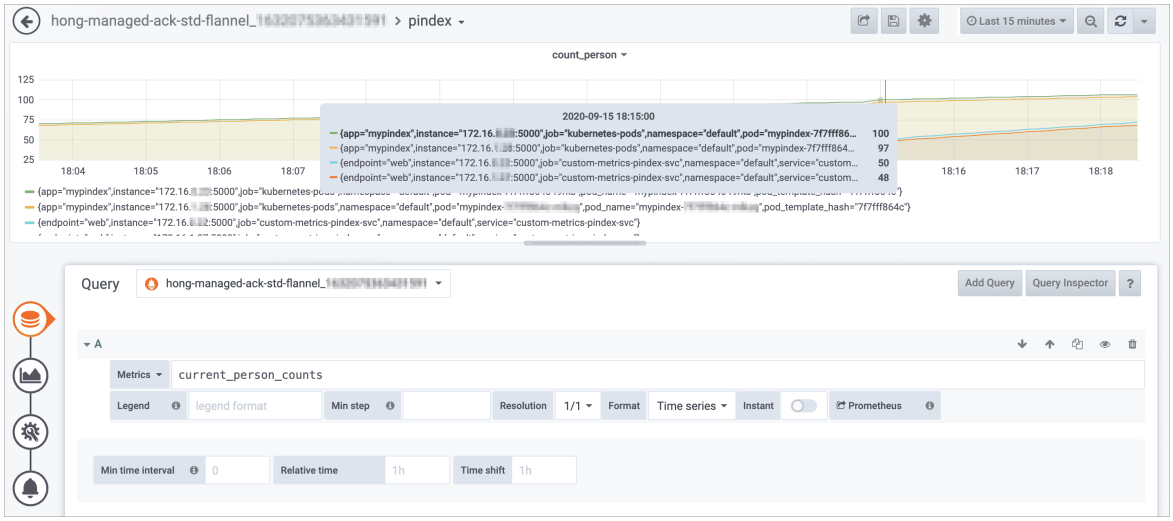
> **Note** The definition of a ServiceMonitor provides more information than an annotation, and includes the namespace and name of the Service.

7. Access the public IP address of the Service that is created in Step 5. This increases the value of the following custom metric.

For more information about metrics, see Data model.



8. Go to the Dashboards page of the ARMS console and click a dashboard to go to the Grafana page. Click **Add panel** in the upper-right corner, select the **Graph** type, and then enter *current_person_counts* for **Metrics**.
9. Save the settings to view the Grafana chart of the custom metric.



FAQ

What can I do if I fail to reinstall ARMS Prometheus after I delete the namespace of ARMS Prometheus?

If you delete only the namespace of ARMS Prometheus, resource configurations may be retained. In this case, you may fail to reinstall ARMS Prometheus. You can perform the following operations to delete the resource configurations:

1. Run the following commands to delete the related ClusterRoles:

```
kubectl delete ClusterRole arms-kube-state-metrics
kubectl delete ClusterRole arms-node-exporter
kubectl delete ClusterRole arms-prom-ack-arms-prometheus-role
kubectl delete ClusterRole arms-prometheus-oper3
kubectl delete ClusterRole arms-prometheus-ack-arms-prometheus-role
kubectl delete ClusterRole arms-pilot-prom-k8s
```

2. Run the following commands to delete the related ClusterRoleBindings:

```
kubectl delete ClusterRoleBinding arms-node-exporter
kubectl delete ClusterRoleBinding arms-prom-ack-arms-prometheus-role-binding
kubectl delete ClusterRoleBinding arms-prometheus-oper-bind2
kubectl delete ClusterRoleBinding kube-state-metrics
kubectl delete ClusterRoleBinding arms-pilot-prom-k8s
kubectl delete ClusterRoleBinding arms-prometheus-ack-arms-prometheus-role-binding
```

3. Run the following commands to delete the related Roles and RoleBindings:

```
kubectl delete Role arms-pilot-prom-spec-ns-k8s
kubectl delete Role arms-pilot-prom-spec-ns-k8s -n kube-system
kubectl delete RoleBinding arms-pilot-prom-spec-ns-k8s
kubectl delete RoleBinding arms-pilot-prom-spec-ns-k8s -n kube-system
```

Related information

•

11.3.6. Ingress Dashboard

Ingress controllers of Container Service for Kubernetes (ACK) allow you to stream all HTTP request log to standard outputs. ACK is also integrated with Log Service. You can create dashboards to monitor and analyze log data. This topic describes how to use Ingress Dashboard with Application Real-Time Monitoring Service (ARMS) to monitor applications.

Prerequisites

You must install the Log Service component before you can use Ingress Dashboard to monitor applications. You can use the following methods to install the component:

- When you create an ACK cluster, select **Enable Log Service**, **Install Ingress Controllers**, and **Create Ingress Dashboard**.
- To install the component for an existing ACK cluster, go to the Add-ons page. For more information, see Collect log files from containers by using Log Service.

Step 1: View the Ingress access log on dashboards

1. Log on to the Log Service console.
2. In the **Projects** section, click the name of the project that you specified when you created the ACK cluster. The details page of the project appears. By default, a project that is named in the format of **k8s-log-{cluster-id}** is created for the ACK cluster.
3. On the page that appears, click the icon on the left side of **nginx-ingress** on the **Logstores** tag.

   > **Note** All of the Ingress access logs are stored in the **nginx-ingress** Logstore.

4. Click **Visual Dashboard** below **nginx-ingress** to view the dashboards of all Ingresses.

   Ingress Dashboard contains five preset charts: Ingress Overview, Ingress Exceptions Center, Ingress Access Center, Ingress Monitoring Center for Blue/Green Deployment, and Ingress Monitoring Center. The following section introduces the Ingress Overview and Ingress Monitoring Center for Blue/Green Deployment charts. For more information about other preset charts, see Monitor nginx-ingress and analyze the access log of nginx-ingress.

   Ingress Overview

   The Ingress Overview dashboard displays the data of a website that is monitored by nginx-ingress. You can view the following information:

   - Website data of the last 24 hours, including the number of page views (PVs), the number of unique visitors (UVs), inbound and outbound traffic, the average latency, the proportion of mobile users, and the proportions of 5xx errors and 404 errors.
   - Website data of the last one minute, including the number of PVs, the number of UVs, the success rate of requests, the proportion of 5xx errors, the average latency, the P95 latency, and the P99 latency.
   - Detailed information about requests in the last 24 hours, including the PV trend (compared with the PV trend in the last seven days), regional distribution of request sources, the top 10 source areas and cities, the proportion of mobile users, and the proportions of Android users and iOS users.
   - Top 10 URLs in the last one hour, including the 10 URLs of the highest PVs, the 10 URLs of the highest latencies, the 10 URLs that return the most 5xx errors, and the 10 URLs that return the most 404 errors.

   Ingress Monitoring Center for Blue/Green Deployment

The Ingress monitoring center for blue/green deployment dashboard displays the real-time status of a service version release and compares the specified service versions. This allows you to identify exceptions and roll back the service at the earliest opportunity. You must set **ServiceA** and **ServiceB** for monitoring and comparison. The dashboard displays the following dynamic monitoring data of each service: the number of PVs, the proportion of 5xx errors, the success rate of requests, the average latency, the P95, P99, and P9999 latencies, and the traffic data.

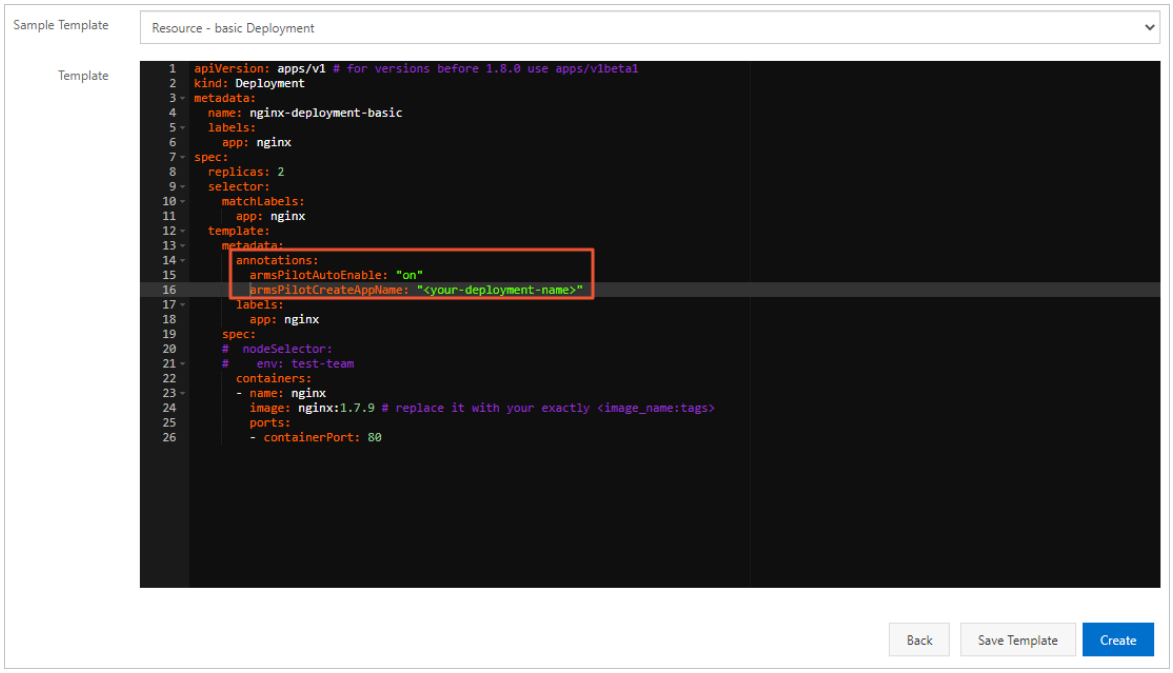
Step 2: Enable ARMS for Java applications

To enable ARMS when you create an application, perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. On the page that appears, select a template from the **Sample Template** drop-down list and add the following `annotations` to the *spec > template > metadata* section in the **template**.

> **Note** Replace *<your-deployment-name>* with the name of your application.

```
annotations:
 armsPilotAutoEnable: "on"
 armsPilotCreateAppName: "<your-deployment-name>"
```



The following YAML template shows how to create a stateless application and enable ARMS for the application:

Show all content of the YAML file (Java) ▼

Step 3: Create an Ingress for the application

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Network > Ingresses**.
5. On the **Ingresses** page, click **Create** in the upper-right corner.
6. In the **Create** dialog box, configure the Ingress and click **Create**.

   For more information about how to configure an Ingress, see Basic operations of an Ingress.

Step 4: Use Ingress Dashboard with ARMS

1. Log on to the Log Service console.
2. In the **Projects** section, click the name of the project that you specified when you created the ACK cluster. The details page of the project appears. By default, a project that is named in the format of **k8s-log-{cluster-id}** is created for the ACK cluster.
3. On the details page of the project, you are redirected to the Logstores tab. All of the Ingress access logs are stored in the **nginx-ingress** Logstore. In the left-side navigation pane, click **Visual Dashboards**.
4. In the dashboard list, click **Ingress Overview V1.2**.
5. On the **Ingress Overview V1.2** page, click **Top 10 Request URLs by Latency** to view the top 10 URLs with the highest latency.
6. On the **Ingress Overview V1.2** page, click **URL (ARMS)** to view the monitoring data about the corresponding Service. You can view the trace details of a specific Service in the ARMS console.

11.3.7. Implement network observability by using ACK Terway Hubble

Container Service for Kubernetes (ACK) Terway Hubble is a network architecture, workload, and topology observability platform. You can deploy ACK Terway Hubble in a managed Kubernetes cluster and then view network traffic and network policies in ACK Terway Hubble. This topic describes how to use the network observability of ACK Terway Hubble to view statistics about network traffic in a container network, such as the sources and destinations of packets.

Prerequisites

创建Kubernetes托管版集群

> **Note** ACK Terway Hubble supports only the One ENI for Multi-Pod mode of Terway. This mode is based on IPvlan. Therefore, when you create the managed Kubernetes cluster, you must specify **Terway** as **Network Plug-in** and **IPvlan** as **Terway Mode**. Otherwise, you cannot use ACK Terway Hubble.

Step 1: Modify the Terway ConfigMap eni-config

Modify the Terway ConfigMap eni-config in the console

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.

3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. Modify the Terway ConfigMap eni-config.
   i. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
   ii. On the top of the **ConfigMap** page, select kube-system as **Namespace**, and click **Edit YAML** in the **Actions** column for eni-config.
   iii. In the **View in YAML** panel, find `10-terway.conf`. Then, set the following parameters under `10-terway.conf` and click **OK**.

```
"cilium_enable_hubble":"true",
"cilium_hubble_listen_address":":4244",
"cilium_hubble_metrics_server":":9091",
"cilium_hubble_metrics":"drop,tcp,flow,port-distribution,icmp",
```

Parameter	Description	Remarks
eniip_virtual_type	Specifies whether to enable the IPvlan mode.	If the ConfigMap does not contain this parameter or the value is not set to `IPVLAN`, your cluster does not support ACK Tereway Hubble.
cilium_enable_hubble	Specifies whether to enable ACK Terway Hubble to analyze network traffic.	This parameter is set to `"true"` in this example.
cilium_hubble_listen_address	Specifies the port that is used to expose the Hubble Service.	This parameter is set to `":4244"` in this example.
cilium_hubble_metrics_server	Specifies the port that is used to expose the Hubble metrics server.	This parameter is set to `":9091"` in this example.
cilium_hubble_metrics	Separate the Hubble metrics with commas (,).	Layer 7 network capabilities such as HTTP and DNS are not supported. ACK Terway Hubble can collect the following metrics: `"drop,tcp,flow,port-distribution,icmp"`. **Note** If you specify excessive metrics, the performance of ACK Terway Hubble may be degraded.

5. Restart the Terway pods for the modified ConfigMap to take effect.
   i. In the left-side navigation pane of the details page, choose **Workloads > Pods**.
   ii. On the top of the **ConfigMap** page, select kube-system as **Namespace**. Then, enter terway-eniip into the search box, and click **Delete** in the **Actions** column for terway-eniip-xxx.

iii. In the **Delete Pod** message, click **OK**.
On the **Pods** page, if the **Status** column of terway-eniip-xxx displays **Running**, this indicates that the pod is restarted.

iv. Repeat the preceding steps to delete all Terway pods.

Modify the Terway ConfigMap eni-config by using the CLI

1. Connect to an ACK cluster by using kubectl.
2. Modify the Terway ConfigMap eni-config.
   i. Run the following command to modify the Terway ConfigMap eni-config:

```
kubectl -n kube-system edit configmap eni-config
```

   ii. Paste the following content to the Terway ConfigMap and save the ConfigMap:

```
"cilium_enable_hubble":"true",
"cilium_hubble_listen_address":":4244",
"cilium_hubble_metrics_server":":9091",
"cilium_hubble_metrics":"drop,tcp,flow,port-distribution,icmp",
```

Parameter	Description	Remarks
eniip_virtual_type	Specifies whether to enable the IPvlan mode.	If the ConfigMap does not contain this parameter or the value is not set to `IPVLAN`, your cluster does not support ACK Tereway Hubble.
cilium_enable_hubble	Specifies whether to enable ACK Terway Hubble to analyze network traffic.	This parameter is set to `"true"` in this example.
cilium_hubble_listen_address	Specifies the port that is used to expose the Hubble Service.	This parameter is set to `":4244"` in this example.
cilium_hubble_metrics_server	Specifies the port that is used to expose the Hubble metrics server.	This parameter is set to `":9091"` in this example.
cilium_hubble_metrics	Separate the Hubble metrics with commas (,).	Layer 7 network capabilities such as HTTP and DNS are not supported. ACK Terway Hubble can collect the following metrics: `"drop,tcp,flow,port-distribution,icmp"`. **Note** If you specify excessive metrics, the performance of ACK Terway Hubble may be degraded.

3. Restart the Terway pods for the modified ConfigMap to take effect.

i. Run the following command to query the Terway pods:

```
kubectl -n kube-system get pod | grep terway-eniip
```

ii. Run the following command to delete a Terway pod:

```
kubectl -n kube-system delete pod terway-eniip-xxx
```

Replace terway-eniip-xxx with the name of the Terway pod. Repeat the preceding steps to delete all Terway pods.

Step 2: Install ACK Terway Hubble

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, search for ack-terway-hubble and click it.
4. Select your cluster in the right-side **Deploy** panel.
5. On the **App Catalog - ack-terway-hubble** page, click the **Parameters** tab. Set `hosts` under `ingress`. The host is used to log on to Hubble UI. You can set other parameters as needed.

   > **Note** The following parameters must be specified under `ingress`.

Parameter	Description
enabled	Specifies whether to use the Ingress to access Hubble-UI.
annotations	Specifies the annotations of the Ingress.
path	Specifies the root path of the Ingress.
hosts	Specifies the host of the Ingress.
tls	Specifies the TLS settings of the Ingress.

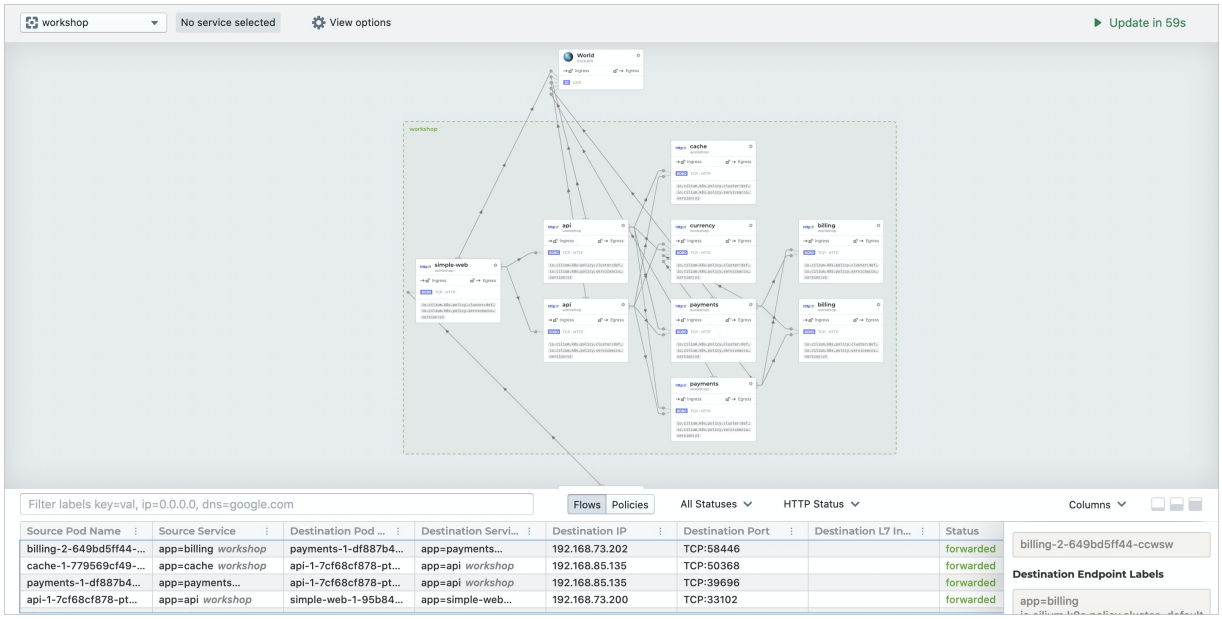
6. In the right-side **Deploy** panel, click **Create**.

Step 3: Get started with ACK Tereway Hubble

If you have set the `hosts` parameter for the Ingress of ACK Terway Hubble, you can log on to Hubble UI by accessing the Ingress host over port 80. When you access the Ingress host, the following information appears:

> **Note** If the domain that you access is not an authoritative domain, such as ingress.local, you must run the `kubectl -n kube-system get svc nginx-ingress-lb` command to query the IP address of Hubble UI. Then, modify the hosts file on your computer to map ingress.local to the IP address of Hubble UI.

- In the upper portion of the page, you can view the topologies of pods and Services that belong to different namespaces.
- In the lower portion of the page, you can view the sources, destinations, ports, and forwarding states of network traffic.
- If you have configured network policies, you can view packets that are dropped because of network policy mismatching.



ACK Terway Hubble uses the hubble-metrics Service in the kube-system namespace to expose network flow metrics. You can specify the metrics that ACK Terway Hubble exposes by setting the `cilium_hubble_metrics` parameter in the Terway ConfigMap eni-config. You can use Prometheus Service and Application Real-Time Monitoring Service (ARMS) Prometheus to collect these metrics. For more information, see Use Prometheus to monitor a Kubernetes cluster and Enable ARMS Prometheus.

> **Note** For more information about the metrics that ACK Terway Hubble can expose, see hubble-exported-metrics.

Metric	Name	Label	Description
drop	hubble_drop_total	reason, protocol	Number of dropped packets.
tcp	hubble_tcp_flags_total	flag, family	TCP flag occurrences.
flow	hubble_flows_processed_total	type, subtype, verdict	Total number of network flows processed.
port-distribution	hubble_port_distribution_total	protocol, port	Numbers of packets distributed by destination port.
icmp	hubble_icmp_total	family, type	Number of ICMP messages

11.3.8. Delete ARMS Prometheus and ACK Prometheus

This topic describes how to uninstall Application Real-Time Monitoring Service (ARMS) Prometheus and Container Service for Kubernetes (ACK) Prometheus.

Procedure

You can uninstall ARMS Prometheus and ACK Prometheus in the ACK console. All data of ARMS Prometheus and ACK Prometheus is uninstalled. The procedures for deleting ARMS Prometheus and ACK Prometheus are similar. In the following example, ARMS Prometheus is uninstalled.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Helm**.
5. On the **Helm** page, find the component whose **Release Name** is arms-prom. Then, click **Delete** in the Actions column.

   > **Note** If you want to uninstall ACK Prometheus, navigate to the **Helm** page and find the component whose **Release Name** is ack-prometheus-operator. Then, click **Delete** in the **Actions** column.

6. In the **Delete** dialog box, select **Clear Release Records** and click **OK**.

11.4. Alert management

Container Service for Kubernetes (ACK) allows you to configure alerts to centrally manage exceptions in the cluster and provides various metrics for different scenarios. By default, the alerting feature is enabled when you create clusters. ACK allows you to deploy Custom Resource Definitions (CRDs) in a cluster to configure and manage alert rules. This topic describes how to set up alerting and grant the cluster the permissions to access alerting resources.

Scenarios

ACK allows you to configure and manage alerts in a centralized manner to monitor various scenarios. The alerting feature is commonly used in the following scenarios:

- **Cluster O&M**
  You can configure alerts to detect exceptions in cluster management, storage, networks, and elastic scaling at the earliest opportunity. For example, you can configure and enable **Alert Rule Set for Node Exceptions** to monitor exceptions in all nodes or specific nodes in the cluster. You can configure and enable **Alert Rule Set for Storage Exceptions** to monitor changes and exceptions in cluster storage. You can configure and enable **Alert Rule Set for Network Exceptions** to monitor changes and exceptions in cluster networks. You can configure and enable **Alert Rule Set for O&M Exceptions** to monitor changes and exceptions in cluster management operations.
- **Application development**
  You can configure alerts to detect exceptions and abnormal metrics of running applications in the cluster at the earliest opportunity. For example, you can configure alerts to detect exceptions of pod replicas and check whether the CPU and memory usage of a Deployment exceed the thresholds. You can use the default alert template to quickly set up alerts to receive notifications about exceptions of pod replicas in the cluster. For example, you can configure and enable **Alert Rule Set for Pod Exceptions** to monitor exceptions in the pods of your application.
- **Application management**
  To monitor the issues that occur throughout the lifecycle of an application, we recommend that you pay attention to application health, capacity planning, cluster stability, exceptions, and errors. You can configure and enable **Alert Rule Set for Critical Events** to monitor warnings and errors in the cluster. You can configure and enable **Alert Rule Set for Resource Exceptions** to monitor resource usage in the cluster and optimize capacity planning.
- **Multi-cluster management**

When you manage multiple clusters, you may find it a complex task to configure and synchronize alert rules across the clusters. ACK allows you to **deploy CRDs in the cluster to manage alert rules**. You can configure the same CRDs to conveniently synchronize alert rules across multiple clusters.

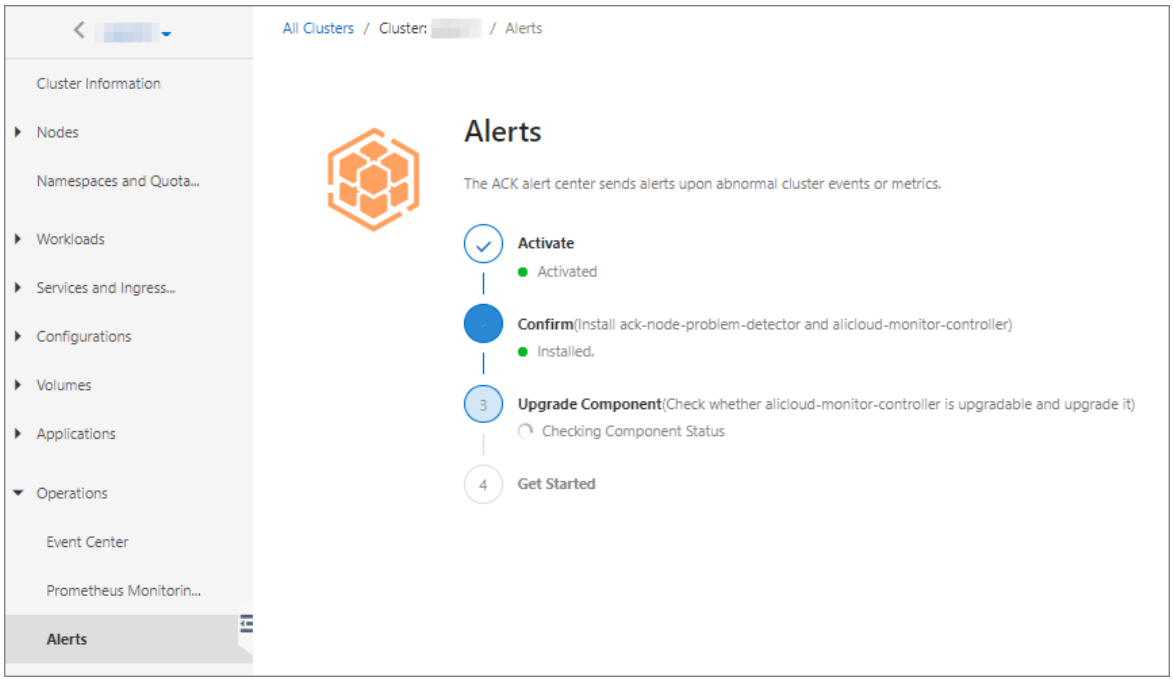
Install and upgrade the components

The console automatically checks whether components need to be activated, installed, or upgraded before you can enable alerting.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Operations > Alerts**.
5. On the **Alerts** page, the console automatically checks whether the following conditions are met.

   If not all conditions are met, follow the on-screen instructions to install or upgrade the required components.

   - Log Service is activated. If you have not activated Log Service, log on to the Log Service console and follow the on-screen instructions to activate the service.
   - **Event Center** is installed. For more information, see Event monitoring.
   - The alicloud-monitor-controller component is upgraded to the latest version. For more information, see alicloud-monitor-controller.



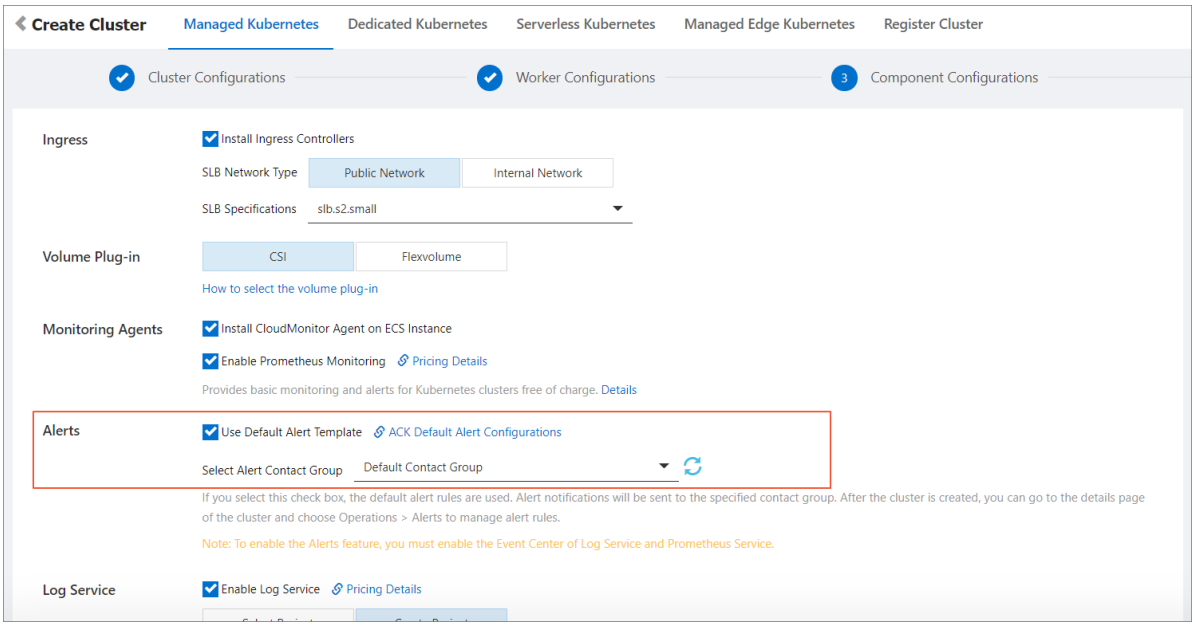
Set up alerting

Managed ACK clusters and dedicated ACK clusters support the alerting feature.

Step 1: Enable the default alert rules

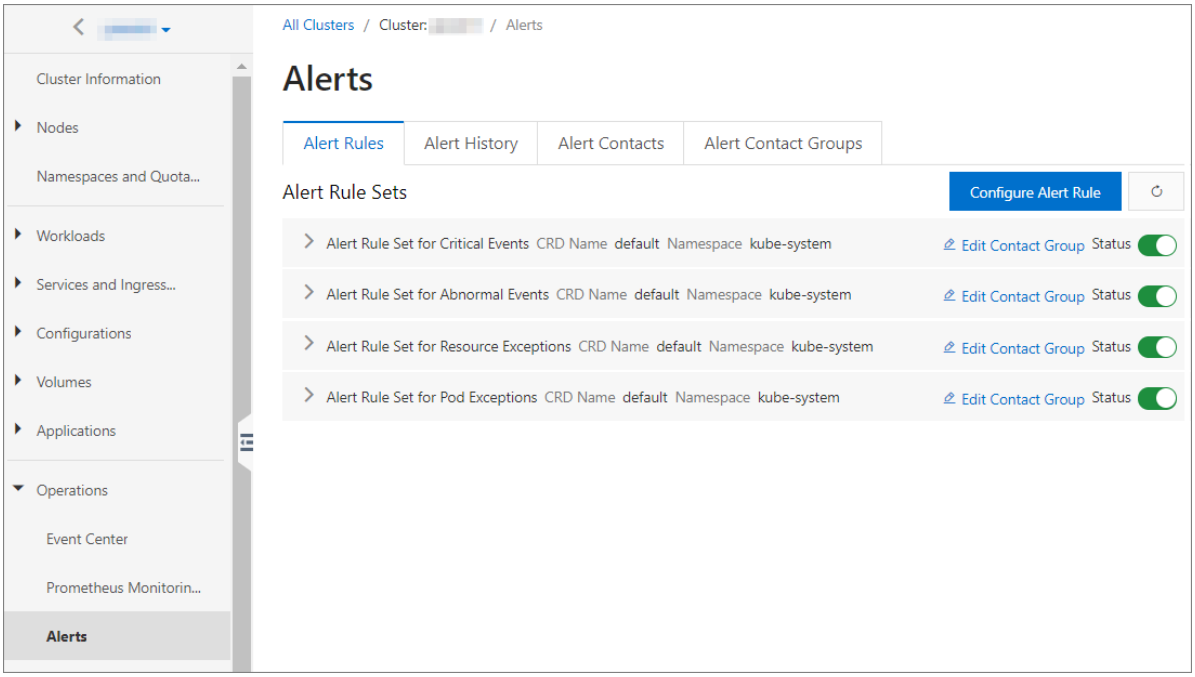
- When you create a managed ACK cluster, select **Use Default Alert Template** and specify an alert contact group.

After this option is selected, the system automatically creates default alert rules and sends alert notifications to the specified contact group.



For more information, see 创建Kubernetes托管版集群.

- To set up alerting for an existing cluster, you can enable alert rules for the cluster.
  i. In the left-side navigation pane, choose **Operations > Alerts**.
  ii. On the **Alert Rules** tab, select an alert rule set and turn on **Status** to enable the alert rule set.



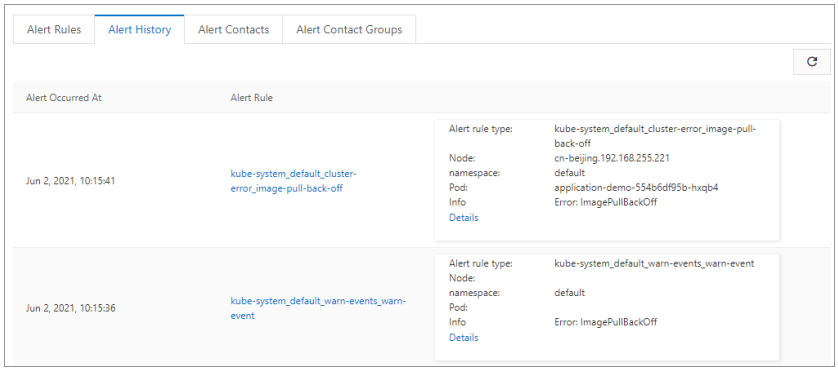
For more information, see Step 2: Configure alert rules

Step 2: Configure alert rules

After you create a managed ACK cluster or dedicated ACK cluster, you can manage alert rules, alert contacts, and alert contact groups.

1. Log on to the ACK console.

2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Operations > Alerts**.

Feature	Description
Alert Rules	○ By default, ACK provides an alert template that is used to generate alerts for exceptions and metrics. ○ Alert rules are classified into several alert rule sets. You can configure multiple alert contact groups for each alert rule set and enable or disable alert rule sets. ○ An alert rule set consists of multiple alert rules, and each alert rule corresponds to an alert item. You can create a YAML file to configure multiple alert rule sets on a cluster. You can also update the YAML file to generate alert rules. ○ For more information about how to configure alert rules by using a YAML file, see Configure alert rules by using CRDs. ○ For more information about the default alert template, see The default alert template.
Alert History	You can view up to 100 historical alerts. You can select an alert and click the link in the **Alert Rule** column to view rule details in the monitoring system. You can click **Details** to go to the resource page where the alert is triggered. The alert may be triggered by an exception or an abnormal metric. 
Alert Contacts	You can create, edit, or delete alert contacts.
Alert Contact Groups	You can create, edit, or delete alert contact groups. When no alert contact group exists, the console automatically creates the default alert contact group based on your registration information.

5. On the **Alert Rules** tab, click **Modify Contacts** to configure the alert contact group to which the alerts are sent. You can turn on or turn off **Status** to enable or disable the alert rule set.

Configure alert rules by using CRDs

When the alerting feature is enabled, the system automatically creates a resource object of the AckAlertRule type in the kube-system namespace. This resource object contains the default alert template. You can use this resource object to configure alert rule sets.

1. Log on to the ACK console.

2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Operations > Alerts**.
5. On the **Alert Rules** tab, click **Configure Alert Rule** in the upper-right corner. You can view the configuration of the AckAlertRule resource object and modify the YAML file to update the configuration.

   The following YAML file is provided as an example of the alert rule configuration:

```
apiVersion: alert.alibabacloud.com/v1beta1
kind: AckAlertRule
metadata:
 name: default
spec:
 groups:
  - name: pod-exceptions              ## The name of the alert rule set.
    rules:
     - name: pod-oom                  ## The name of the alert rule.
       type: event                    ## The type of the alert rule. Valid enumeration values: event and metric.
       expression: sls.app.ack.pod.oom    ## The expression of the alert rule. When the type of the alert rule is s
et to event, the expression must be set to sls_event_id, which is the event ID in Log Service.
       enable: enable                 ## The status of the alert rule. Valid enumeration values: enable and disable.
     - name: pod-failed
       type: event
       expression: sls.app.ack.pod.failed
       enable: enable
```

The default alert template

ACK creates the default alert rules based on the following conditions:

- The default alert rules are enabled.
- You go to the Alert Rules tab for the first time and default alert rules are not enabled.

The following table describes the default alert rules.

Alert Rule Set	Alert Rule	ACK_CR_Rule_Name	SLS_Event_ID
Alert Rule Set for Critical Events	Errors	error-event	sls.app.ack.error
	Warnings	warn-event	sls.app.ack.warn
Alert Rule Set for Node Exceptions	Docker process exceptions on nodes	docker-hang	sls.app.ack.docker.hang
	Evictions	eviction-event	sls.app.ack.eviction
	GPU XID errors	gpu-xid-error	sls.app.ack.gpu.xid_error
	Node restarts	node-restart	sls.app.ack.node.restart
	Network Time Protocol (NTP) service failures on nodes	node-ntp-down	sls.app.ack.ntp.down

Alert Rule Set	Alert Rule	ACK_CR_Rule_Name	SLS_Event_ID
	Pod Lifecycle Event Generator (PLEG) errors on nodes	node-pleg-error	sls.app.ack.node.pleg_error
	Process errors on nodes	ps-hang	sls.app.ack.ps.hang
Alert Rule Set for Resource Exceptions	Excess file handles on nodes	node-fd-pressure	sls.app.ack.node.fd_pressure
	Insufficient node disk space	node-disk-pressure	sls.app.ack.node.disk_pressure
	Excessive processes on nodes	node-pid-pressure	sls.app.ack.node.pid_pressure
	Insufficient node resources for scheduling	node-res-insufficient	sls.app.ack.resource.insufficient
	Insufficient node IP addresses	node-ip-pressure	sls.app.ack.ip.not_enough
Alert Rule Set for Pod Exceptions	Pod out-of-memory (OOM) errors	pod-oom	sls.app.ack.pod.oom
	Pod restart failures	pod-failed	sls.app.ack.pod.failed
	Image pull failures	image-pull-back-off	sls.app.ack.image.pull_back_off
	No available Server Load Balancer (SLB) instance	slb-no-ava	sls.app.ack.ccm.no_ava_slb
	SLB instance update failures	slb-sync-err	sls.app.ack.ccm.sync_slb_failed
	SLB instance deletion failures	slb-del-err	sls.app.ack.ccm.del_slb_failed
	Node deletion failures	node-del-err	sls.app.ack.ccm.del_node_failed
	Node addition failures	node-add-err	sls.app.ack.ccm.add_node_failed
	Route creation failures	route-create-err	sls.app.ack.ccm.create_route_failed
	Route update failures	route-sync-err	sls.app.ack.ccm.sync_route_failed
	High-risk configurations detected in inspections	si-c-a-risk	sls.app.ack.si.config_audit_high_risk

Alert Rule Set	Alert Rule	ACK_CR_Rule_Name	SLS_Event_ID
Alert Rule Set for O&M Exceptions	Command execution failures in managed node pools	nlc-run-cmd-err	sls.app.ack.nlc.run_command_fail
	No command provided in managed node pools	nlc-empty-cmd	sls.app.ack.nlc.empty_task_cmd
	URL mode not implemented in managed node pools	nlc-url-m-unimp	sls.app.ack.nlc.url_mode_unimpl
	Unknown repair operations in managed node pools	nlc-opt-no-found	sls.app.ack.nlc.op_not_found
	Node draining and removal failures in managed node pools	nlc-des-node-err	sls.app.ack.nlc.destroy_node_fail
	Node draining failures in managed node pools	nlc-drain-node-err	sls.app.ack.nlc.drain_node_fail
	Elastic Compute Service (ECS) restart timeouts in managed node pools	nlc-restart-ecs-wait	sls.app.ack.nlc.restart_ecs_wait_fail
	ECS restart failures in managed node pools	nlc-restart-ecs-err	sls.app.ack.nlc.restart_ecs_fail
	ECS reset failures in managed node pools	nlc-reset-ecs-err	sls.app.ack.nlc.reset_ecs_fail
	Auto-repair task failures in managed node pools	nlc-sel-repair-err	sls.app.ack.nlc.repair_fail
Alert Rule Set for Network Exceptions	Invalid Terway resources	terway-invalid-res	sls.app.ack.terway.invalid_resource
	IP allocation failures of Terway	terway-alloc-ip-err	sls.app.ack.terway.alloc_ip_fail
	Ingress bandwidth configuration parsing failures	terway-parse-err	sls.app.ack.terway.parse_fail
	Network resource allocation failures of Terway	terway-alloc-res-err	sls.app.ack.terway.allocate_failure
	Network resource reclaim failures of Terway	terway-dispose-err	sls.app.ack.terway.dispose_failure
	Terway virtual mode changes	terway-virt-mod-err	sls.app.ack.terway.virtual_mode_change

Alert Rule Set	Alert Rule	ACK_CR_Rule_Name	SLS_Event_ID
	Pod IP checks executed by Terway	terway-ip-check	sls.app.ack.terway.config_check
	Ingress configuration reload failures	ingress-reload-err	sls.app.ack.ingress.err_reload_nginx
Alert Rule Set for Storage Exceptions	Disk size is less than 20 GiB	csi_invalid_size	sls.app.ack.csi.invalid_disk_size
	Subscription disks cannot be mounted	csi_not_portable	sls.app.ack.csi.disk_not_portable
	Unmount failures occur because the mount target is in use	csi_device_busy	sls.app.ack.csi.deivce_busy
	No disks are available	csi_no_ava_disk	sls.app.ack.csi.no_ava_disk
	I/O hangs of cloud disks	csi_disk_iohang	sls.app.ack.csi.disk_iohang
	Slow I/O of the underlying disks of persistent volume claims (PVCs)	csi_latency_high	sls.app.ack.csi.latency_too_high
	Disk usage exceeds the threshold	disk_space_press	sls.app.ack.csi.no_enough_disk_space

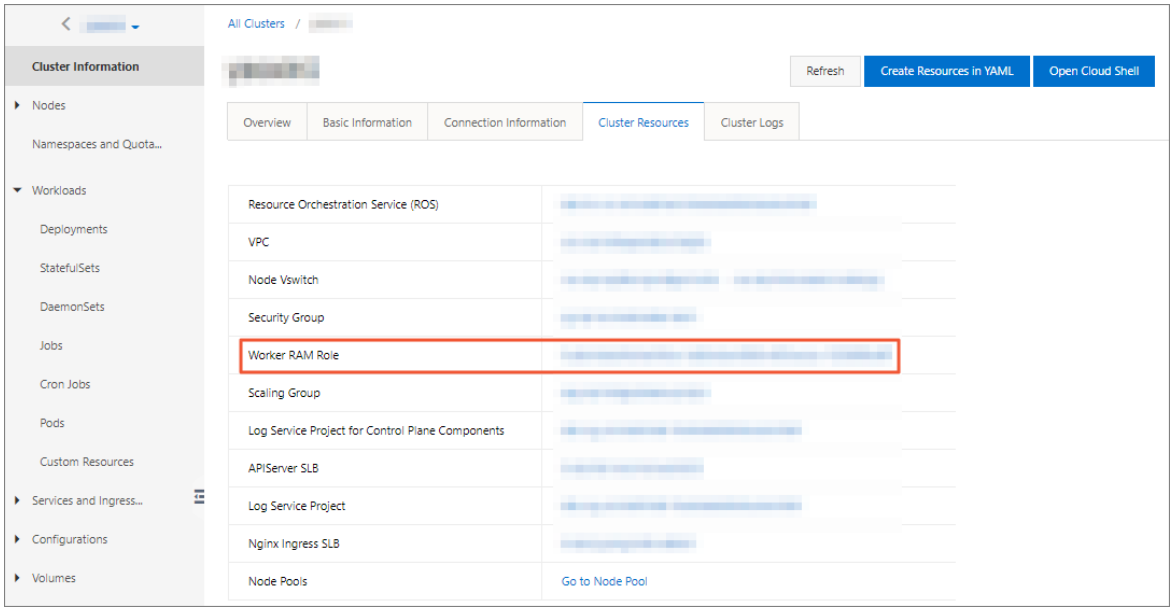
Grant a dedicated cluster the permissions to access alerting resources

Before you can enable the alerting feature in a dedicated cluster, you must grant the required permissions to the cluster.

> **Note** The system automatically grants managed clusters the permissions to access alerting resources in Log Service.

Grant a dedicated cluster the permissions to access alerting resources in Log Service and Application Real-Time Monitoring Service (ARMS) Prometheus. For more information, see Use custom policies to grant permissions to a RAM user and Overview.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the **Cluster Information** page, click the **Cluster Resources** tab and click the link to the right of **Worker RAM Role** to go to the **RAM console**.



5. On the **RAM Roles** page, click the **Permissions** tab. Select the RAM role and click the link in the **Policy** column.
6. On the **Policy Document** tab, click **Modify Policy Document**. In the **Modify Policy Document** panel, copy the following content to the Policy Document field.

```
{
      "Action": [
        "log:*",
        "arms:*",
        "cs:UpdateContactGroup"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
}
```

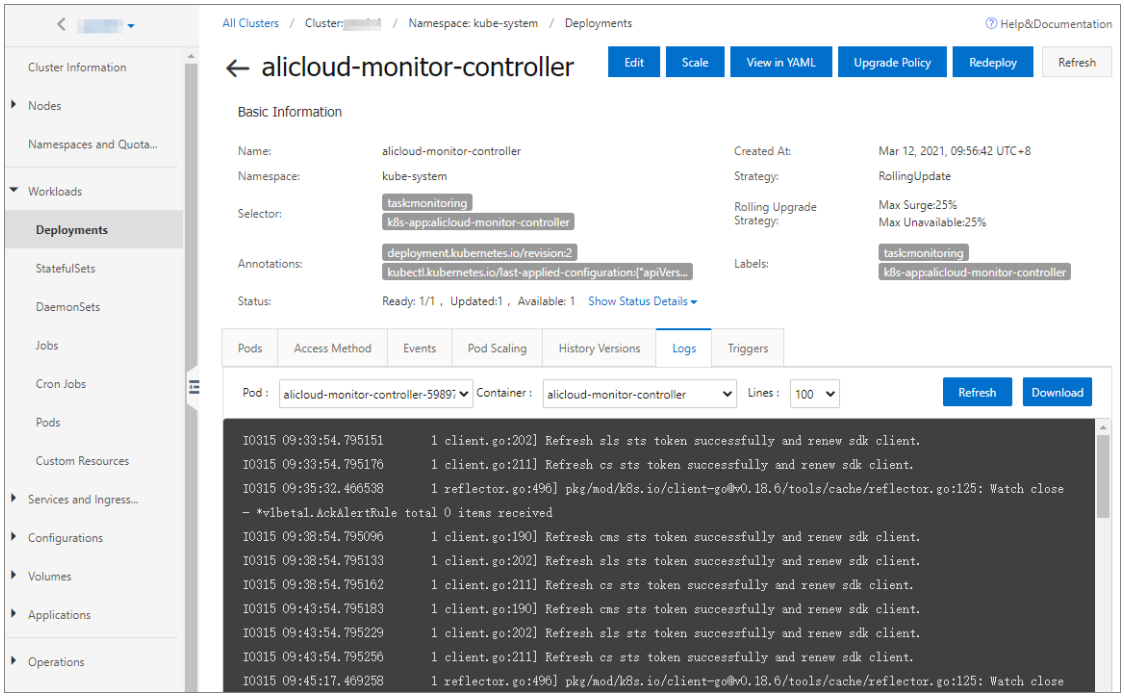
7. Click **OK**.

Verification

i. In the left-side navigation pane of the Container Service for Kubernetes (ACK) console, choose **Workloads > Deployments**.

ii. Select the kube-system **namespace**, find alicloud-monitor-controller in the Deployments list, and then click the link in the **Name** column.

iii. Click the **Logs** tab and verify that the records of successful authorization are displayed.



12.Cost analysis

12.1. Cost analysis

The cost analysis feature of Container Service for Kubernetes (ACK) provides analysis on resource usage and cost distribution. This provides suggestions on how to improve resource utilization and reduce the cost. Cost analysis is suitable for Kubernetes clusters in an enterprise. This topic introduces the cost analysis feature and how to configure cost analysis.

Prerequisites

- An ACK cluster of Kubernetes 1.18.8 or later is created. For more information, see 创建Kubernetes托管版集群.
- Prometheus Service is enabled for the created cluster. For more information, see Enable ARMS Prometheus.

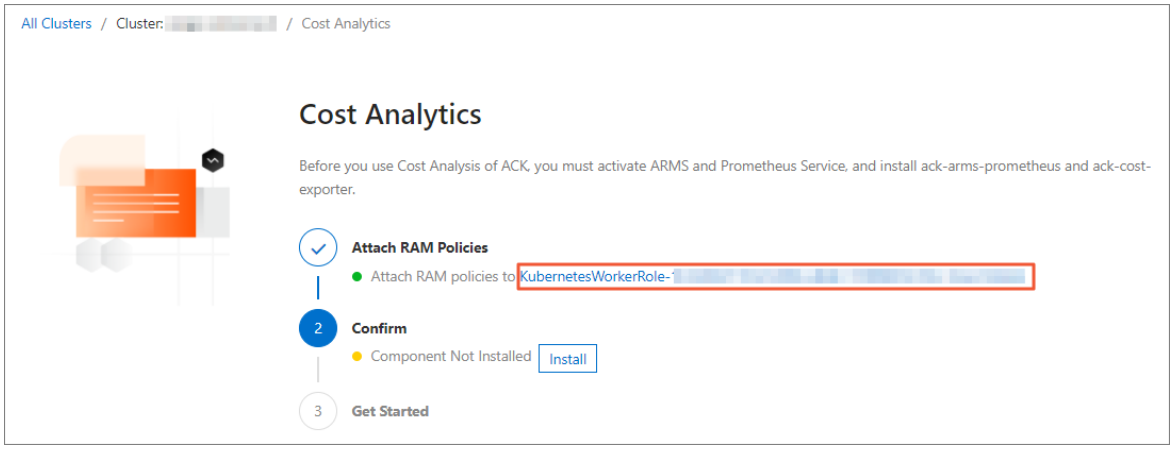
Features

The following features are provided:

- Cost analysis of cloud resources
  Aggregates the costs of all cloud resources within a Kubernetes cluster. Compares the cost of the previous day with that of the same date of the previous year. Compares the cost of the previous day with that of the day before the previous day.
- Cost trend analysis
  Analyzes the cost trends of all cloud resources. Lists cloud resources with the highest cost contributions. This provides information that helps you reduce costs.
- Suggestions on cost saving
  Analyzes the cost trend of a Kubernetes cluster. Provides correlation analysis between resource usage and cost contribution. Provides suggestions to reduce costs.
- Real-time cost forecast
  Simulates cost calculation based on the real-time prices of all computing resources within a cluster. Provides hourly cost forecasts. Multiple billing methods, such as subscription, pay-as-you-go, and preemptible instances, are supported.
- Cost allocation based on namespaces
  Allocates the cost of a cluster based on namespaces. Provides the real-time cost forecast, cost trend, and cost contribution of each namespace.
- Optimization of application costs
  Lists the top 10 applications with the highest cost contributions. Provides the cost trends of the top 10 applications. This provides information that helps you improve the cost-effectiveness of an application.

Configure cost analysis

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Cost Analysis**.
5. On the **Cost Analysis** page, click **KubernetesWorkerRole-***** to go the **RAM Roles** page and configure Resource Access Management (RAM) policies for the RAM role.



i. On the **RAM Roles** page, click the **k8sWorkerRolePolicy-*** policy.

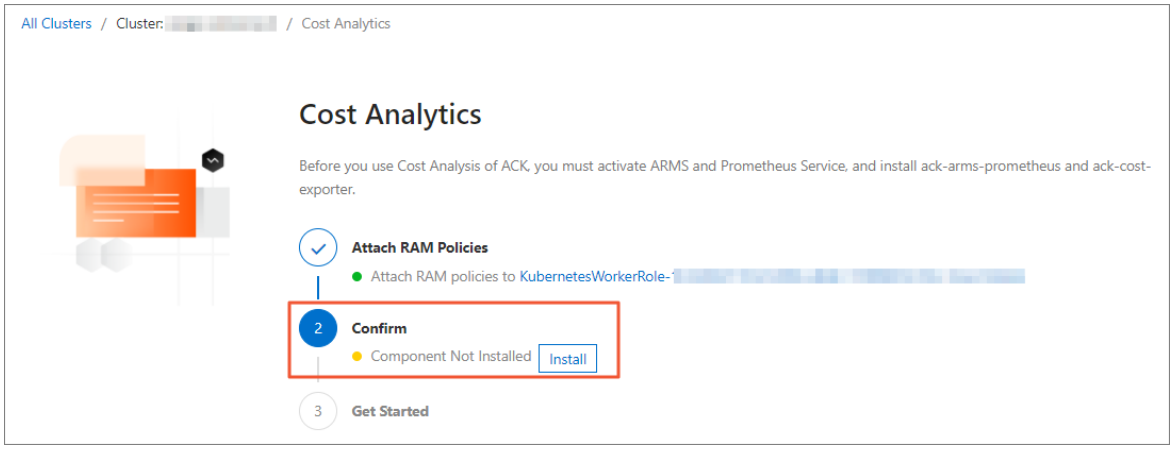
ii. On the **Policies** page, click **Modify Policy Document**.

iii. In the **Modify Policy Document** panel, add the following content and click **OK**.

```
{
  "Action": [
    "bssapi:QueryInstanceBill"
  ],
  "Resource": "*",
  "Effect": "Allow"
},
{
  "Action": [
    "ecs:DescribeSpotPriceHistory",
    "ecs:DescribeInstances",
    "ecs:DescribePrice"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
```

> **Note** Add a comma (,) to the end of an action content before you enter the next action content.

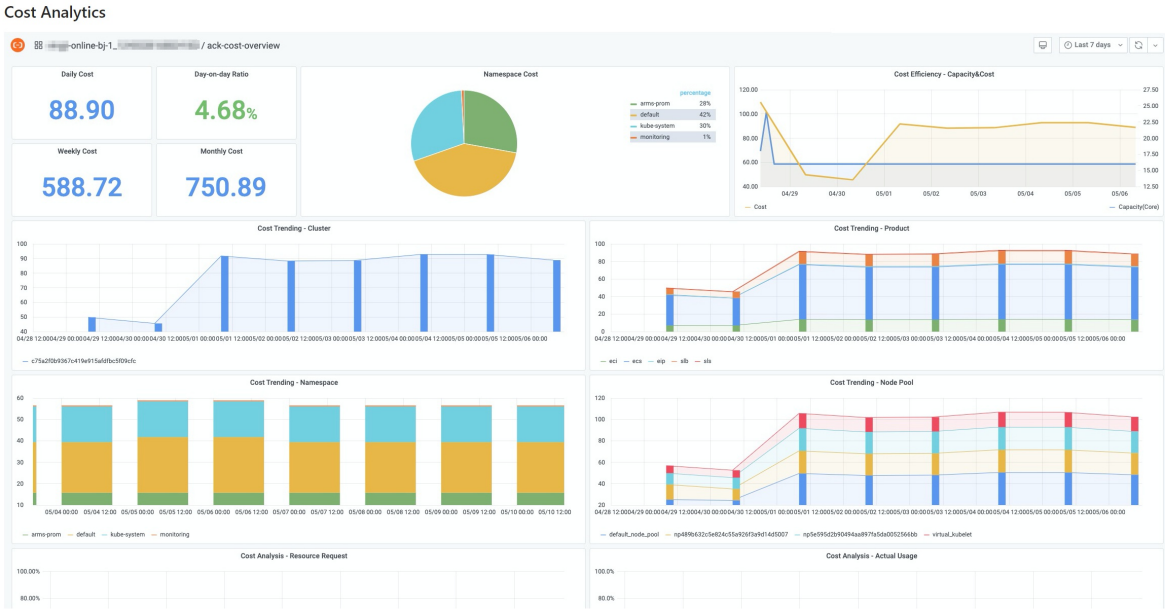
6. On the **Cost Analysis** page, click **Install**.



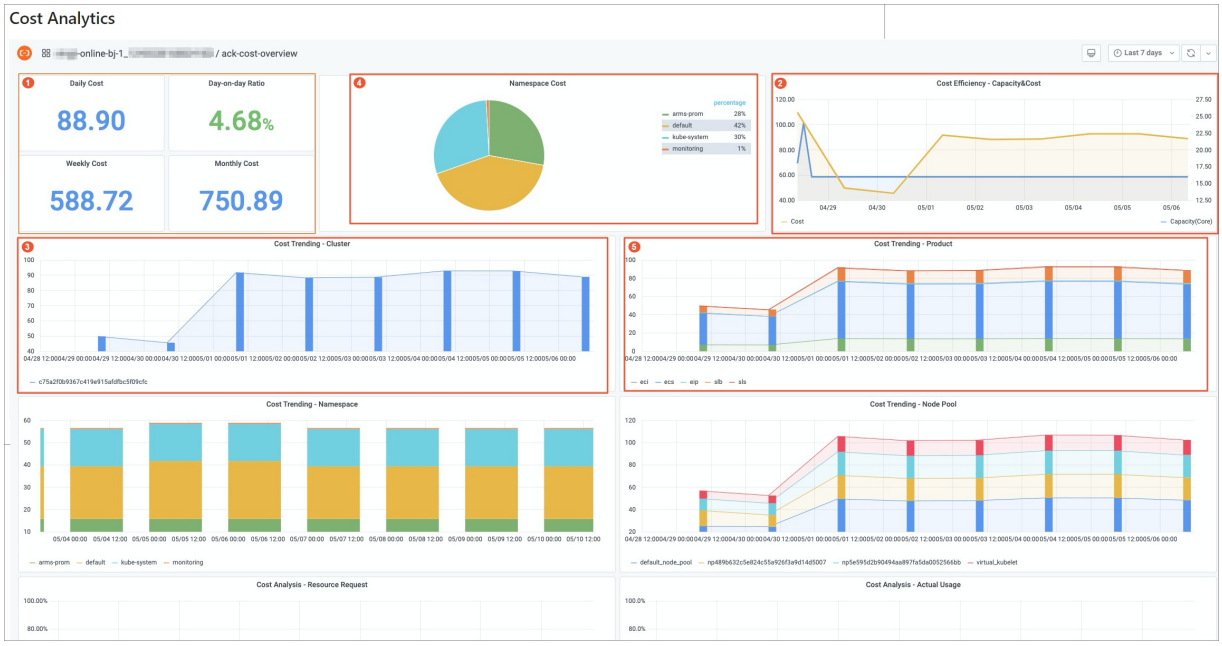
7. On the **Cost Analysis** page, you can view a dashboard that provides visualized information about the

cost.

> **Note** After cost analysis is enabled, the bills are displayed at 08:00:00 (UTC+8) on the next day.



Use cost analysis



The following table describes the features of the dashboard.

No.	Feature	Description

No.	Feature	Description
①	Analyze the resource waste of the cluster based on the cost trend	This figure shows the details of the cost. Daily Cost indicates the cost of the previous day. Day-on-day Ratio indicates the growth rate of the cost of the previous day. If the growth rate is in green color, it indicates that the cost of the previous day is reduced compared with the cost of the day before the previous day. If the growth rate is in red color, it indicates that the cost of the previous day is increased compared with the cost of the day before the previous day
②		The yellow curve indicates the resource consumption. The blue curve indicates the actual resource capacity of the cluster. In common cases, the two curves may be correlated with each other. If the two curves represent different trends, it indicates changes to the average cost of one CPU core. In this case, you can check whether resources of higher prices are used.
③		The cost trend within a time period.
④	Allocate the cost based on namespaces	A cluster may contain nodes of different specifications and billing methods. When you allocate the cost based on namespaces, you must consider the price differences of nodes that host pods in different namespaces, not only the resource consumption of different namespaces. The cost analysis feature converts the real-time cost of each node by using alibaba-cloud-price-exporter. The cost of a namespace is calculated based on the following formula: **Σ (Pod resource requests/Node capacity) × Node unit price** This provides a precise method to calculate the cost of a namespace. However, discounts, vouchers, and subscriptions may cause the actual cost to be different from the cost calculated based on this formula. To obtain the actual cost of a namespace, you can multiply the actual cost of the cluster by the ratio of the cost of this namespace to the total cost of all namespaces.
⑤	Analyze the costs of cloud resources based on cost trends	A cluster may use multiple cloud resources. The costs of cloud resources vary based on the billing rules and how the cloud resources are used by the cluster. This feature provides the cost trends and contributions of different cloud resources. This provides information that helps you reduce the cost.

FAQ

Issue 1: Why is no data displayed after I enable cost analysis? ❯

Issue 2: Why is the total cost of all namespaces not equal to the actual bills? ❯

13.Auto Scaling

13.1. Overview

Auto Scaling (ESS) is a service that can dynamically scale computing resources to meet your business requirements. This provides a more cost-efficient method to manage your resources.

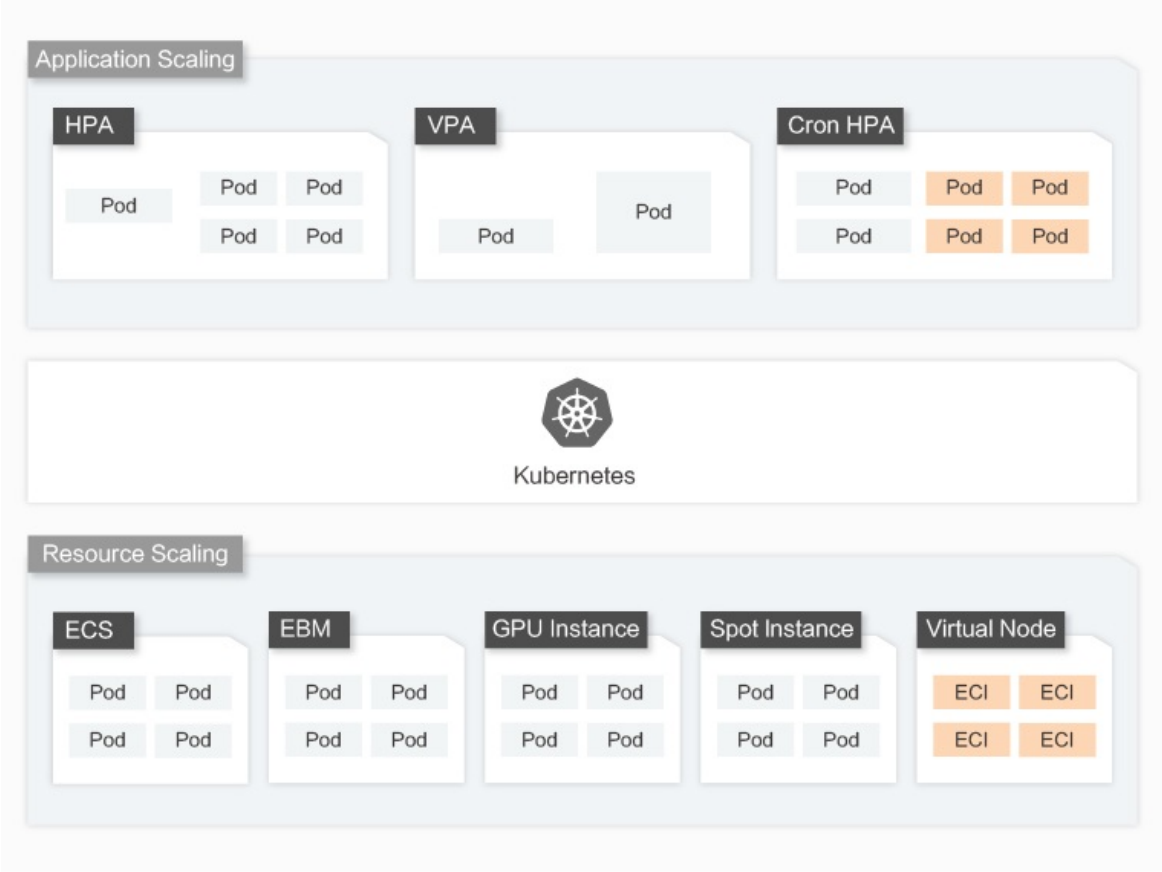
Background information

ESS is widely used in clusters of Container Service for Kubernetes (ACK). Typically, ESS is used in scenarios such as online workload scaling, large-scale computing and training, GPU-accelerated deep learning, inference and training based on shared GPU resources, and periodical workload scheduling. ESS defines elasticity from the following aspects:

- Workload scaling. ESS can adjust workloads, such as pods. For example, Horizontal Pod Autoscaler (HPA) is a typical workload scaling component that can change the number of replicated pods to scale the workload.
- Resource scaling. If the resources of a cluster cannot meet the requirements of the scaled workload, the related component automatically adds Elastic Compute Service (ECS) instances or elastic container instances (ECIs) to the cluster.

The components for workload scaling and resource scaling can be separately used or used in combination. If you want to decouple the components, you must scale the workload within the resource limit of the cluster.

Scaling components for ACK clusters



Components for workload scaling

Component	Description	Scenario	Limit	Reference
HPA	Kubernetes is developed with built-in components. These components are used for online applications.	Online workloads	HPA uses Deployments and StatefulSets to scale workloads.	HPA
Vertical Pod Autoscaler (VPA)	An open source component. VPA is used for monolithic applications.	Monolithic applications	VPA is applicable to applications that cannot be horizontally scaled. In practical scenarios, VPA is used when pods are recovered from anomalies.	Vertical pod autoscaling
CronHPA	An open source component provided by ACK. CronHPA is applicable to applications whose resource usage changes periodically.	Periodically changing workloads	CronHPA uses Deployments and StatefulSets to scale workloads. CronHPA is also compatible with HPA. You can use CronHPA and HPA in combination to scale workloads.	CronHPA
Elastic-Workload	A component provided by ACK. ack-kubernetes-elastic-workload is used to scale workloads with a higher level of precision. For example, it can be applied when workloads are deployed in different zones.	Workloads that require precise scaling	ack-kubernetes-elastic-workload is applicable to online workloads that require precise scaling. For example, some pods of a Deployment are scheduled to an ECS instance, and the remaining pods are scheduled to ECIs.	Install the elastic workload component

Components for resources scaling

Component	Description	Scenario	Time cost for resource deployment	Reference

Component	Description	Scenario	Time cost for resource deployment	Reference
cluster-autoscaler	cluster-autoscaler is an open source component provided by Kubernetes. cluster-autoscaler horizontally scales nodes in a cluster. cluster-autoscaler is integrated with ESS to provide more elastic and cost-efficient scaling services.	cluster-autoscaler is applicable to all scenarios, especially online workloads, deep learning, and large-scale computing.	The time required to add 1,000 nodes to a cluster: • Standard mode: 120 seconds. • Fast mode: 60 seconds. • Standard mode with the Qboot firmware: 90 seconds. • Fast mode with the Qboot firmware: 45 seconds. For more information about Qboot, see Alibaba Cloud Linux 2.1903 64-bit (Quick Start).	Auto scaling of nodes
virtual-node	virtual-node is an open source component provided by ACK. virtual-node provides the runtime for serverless applications. Developers do not need to handle node resources and only need to create, manage, and pay for pods based on the actual usage.	virtual-node is used to handle traffic spikes, continuous integration (CI) and continuous delivery (CD), and big data computing.	The time required to create 1,000 pods: • When image caching is disabled: 30 seconds. • When image caching is enabled: 15 seconds.	Deploy the virtual node controller and use it to create Elastic Container Instance-based pods
virtual-kubelet-autoscaler	virtual-kubelet-autoscaler is a component provided by ACK. virtual-kubelet-autoscaler is used to scale serverless applications.	virtual-kubelet-autoscaler is used to handle traffic spikes, CI and CD, and big data computing.	The time required to create 1,000 pods: • When image caching is disabled: 30 seconds. • When image caching is enabled: 15 seconds.	Install virtual-kubelet-autoscaler

13.2. Auto scaling of nodes

Container Service for Kubernetes (ACK) provides the auto scaling component (cluster-autoscaler) to automatically scale nodes. Regular instances, GPU-accelerated instances, and preemptible instances can be automatically added to or removed from an ACK cluster to meet your business requirements. This component supports multiple scaling modes, various instance types, and instances that are deployed across zones. This component is applicable to diverse scenarios.

How it works

The auto scaling model of Kubernetes is different from the traditional scaling model that is based on the resource usage threshold. Developers must understand the differences between the two scaling models before they migrate workloads from traditional data centers or other orchestration systems, such as Swarm to Kubernetes.

The traditional scaling model is based on resource usage. For example, if a cluster contains three nodes and the CPU utilization or memory usage of the nodes exceeds the scaling threshold, new nodes are added to the cluster. However, you must consider the following issues when you use the traditional scaling model:

1. **How is a resource usage threshold specified and applied?**
2. **How is load balancing applied after instances are added?**
3. **How is a scale-in event triggered and implemented?**

How does the auto scaling model of Kubernetes fix these issues? Kubernetes provides a two-layer scaling model that decouples pod scheduling from resource scaling.

In simple terms, pods are scaled based on resource usage. When pods enter the Pending state due to insufficient resources, a scale-out event is triggered. After new nodes are added to the cluster, the pending pods are automatically scheduled to the newly added nodes. This way, the load of the application is balanced. The following section describes the auto scaling model of Kubernetes in detail:

1. **How are nodes selected during a scale-out event?**
2. **How is a scale-in event triggered?**
3. **How can I select among multiple scaling groups?**
4. **How can I improve the success rate of auto scaling?**
5. **How can I accelerate auto scaling?**

Precaution

- For each account, the default CPU quota for pay-as-you-go instances in each region is 50 vCPUs. You can add at most 48 custom route entries to each route table of a virtual private cloud (VPC). To request a quota increase, submit a ticket.
- The stock of ECS instances may be insufficient for auto scaling if you specify only one ECS instance type for a scaling group. We recommend that you specify multiple ECS instance types with the same specification for a scaling group. This increases the success rate of auto scaling.
- In swift mode, when a node is shut down and reclaimed, the node stops running and enters the *NotReady* state. When a scale-out event is triggered, the state of the node changes to *Ready*.
- If a node is shut down and reclaimed in swift mode, you are charged for only the disks. This rule does not apply to nodes that use local disks, such as the instance type of ecs.d1ne.2xlarge, for which you are also charged a computing fee. If the stock of nodes is sufficient, nodes can be launched within a short period of time.
- If elastic IP addresses (EIPs) are associated with pods, we recommend that you do not delete the scaling group or remove ECS instances from the scaling group in the ECS console. Otherwise, these EIPs cannot be automatically released.

Step 1: Configure auto scaling

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.

3. On the **Clusters** page, follow the instructions to navigate to the **Configure Auto Scaling** page.

   You can navigate to the Configure Auto Scaling page in the following ways:

   - Method 1: Find the cluster that you want to manage and choose **More > Auto Scaling** in the **Actions** column.
   - Method 2:
     a. Find the cluster that you want to manage and click **Details** in the Actions column.
     b. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
     c. In the upper-right corner of the Node Pools page, click **Configure Auto Scaling**.

Step 2: Perform authorization

You must perform authorization in the following scenarios:

If ACK has limited permissions on nodes in the cluster, assign the **AliyunCSManagedAutoScalerRole** Resource Access Management (RAM) role to ACK.

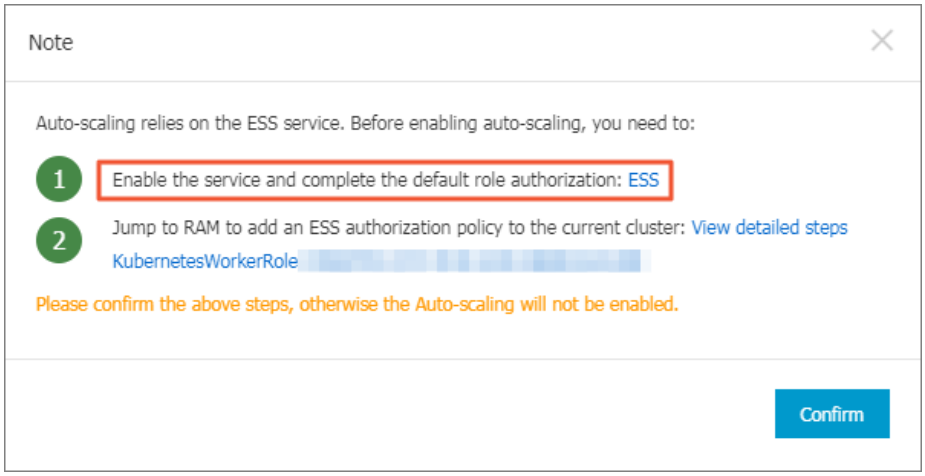
> **Note** You need to perform the authorization only once for each Alibaba Cloud account.

1. Activate Auto Scaling (ESS).
   i. In the dialog box that appears, click the first hyperlink to log on to the **ESS** console.
   ii. Click **Activate Auto Scaling** to go to the **Enable Service** page.
   iii. Select the **I agree with Auto Scaling Agreement of Service** check box and click **Enable Now**.
   iv. On the **Activated** page, click **Console** to log on to the **ESS** console.
   v. Click **Go to Authorize** to go to the **Cloud Resource Access Authorization** page. Then, authorize ESS to access other cloud resources.
   vi. Click **Confirm Authorization Policy**.
2. Assign the RAM role.
   i. Click the second hyperlink in the dialog box.

   > **Note** You must log on to the RAM console by using an Alibaba Cloud account.

   ii. On the **Cloud Resource Access Authorization** page, click **Confirm Authorization Policy**.

1. Activate ESS.
   i. In the dialog box that appears, click the first hyperlink to log on to the **ESS** console.



ii. Click **Activate Auto Scaling** to go to the **Enable Service** page.

iii. Select the **I agree with Auto Scaling Agreement of Service** check box and click **Enable Now**.

iv. On the **Activated** page, click **Console** to log on to the **ESS** console.

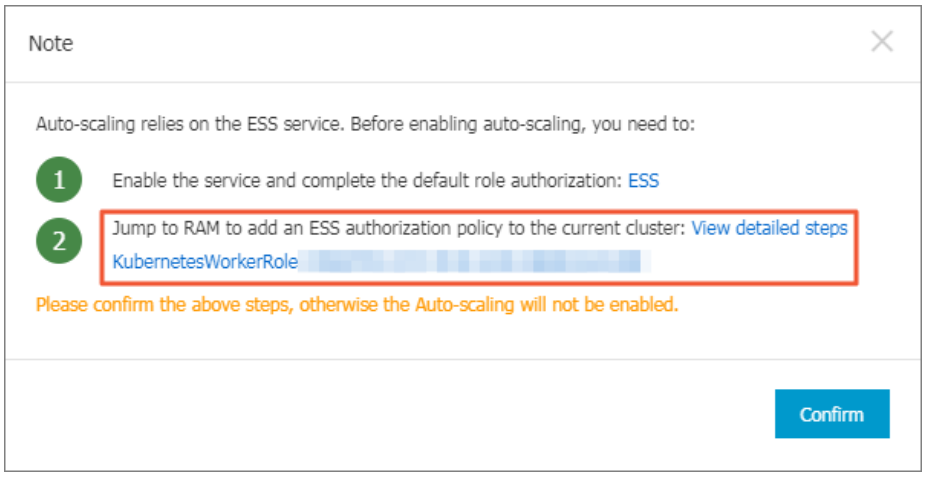
v. Click **Go to Authorize** to go to the **Cloud Resource Access Authorization** page. Then, authorize ESS to access other cloud resources.

vi. Click **Confirm Authorization Policy**.

If the authorization is successful, you are redirected to the ESS console. Close the page and modify the permissions of the worker RAM role.

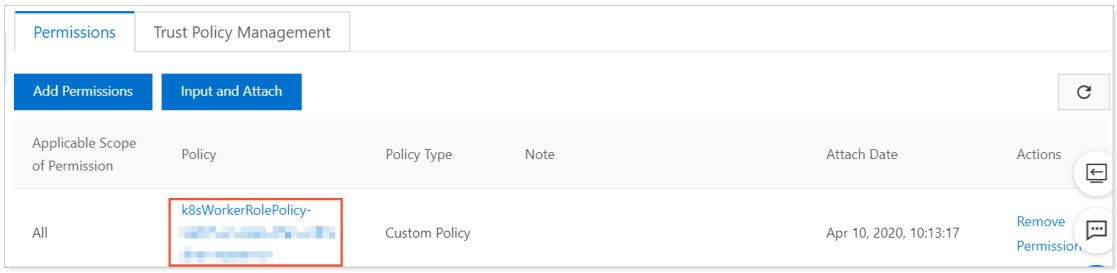
2. Modify the permissions of the worker RAM role.

i. Click the second hyperlink in the dialog box to go to the **RAM Roles** page.

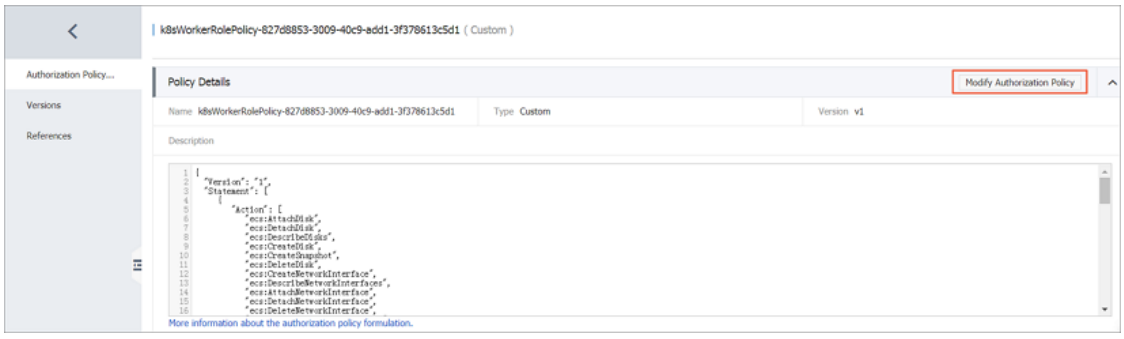


Note You must log on to the RAM console by using an Alibaba Cloud account.

ii. On the **Permissions** tab, click the name of the policy assigned to the RAM role. The details page of the policy appears.



iii. Click **Modify Policy Document**. The **Modify Policy Document** panel appears on the right side of the page.



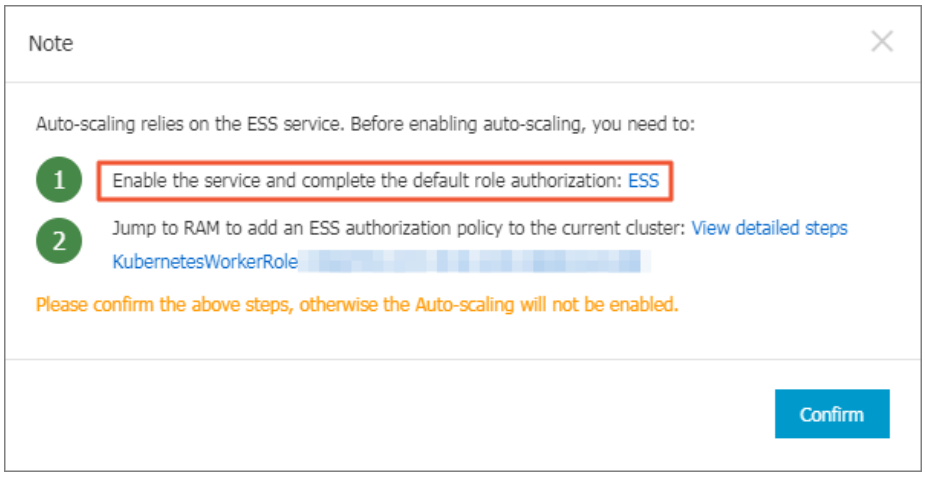
iv. In the **Policy Document** section, add the following policy content to the `Action` field and click **OK**.

```
"ess:Describe*",
"ess:CreateScalingRule",
"ess:ModifyScalingGroup",
"ess:RemoveInstances",
"ess:ExecuteScalingRule",
"ess:ModifyScalingRule",
"ess:DeleteScalingRule",
"ecs:DescribeInstanceTypes",
"ess:DetachInstances",
"vpc:DescribeVSwitches"
```

> **Note** Before you add the policy content, add a comma (,) to the end of the bottom line in the `Action` field.

If you want to associate an auto-scaling group with an EIP, perform the following steps to grant permissions:

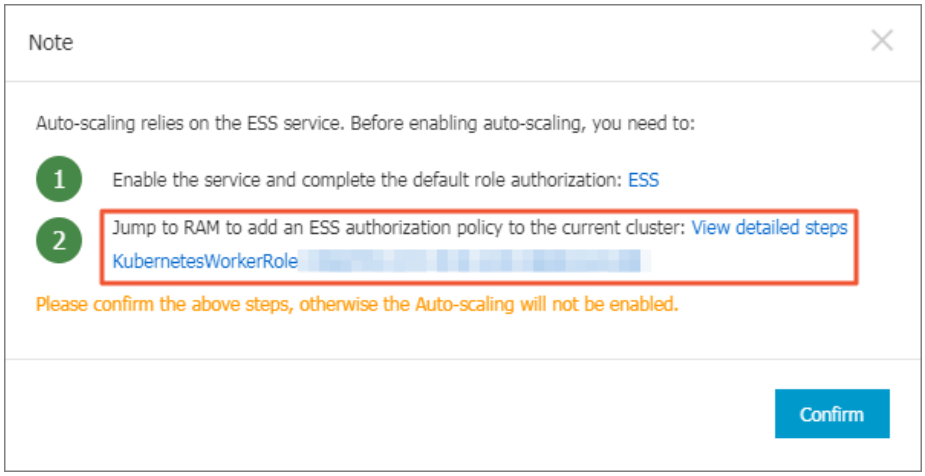
1. Activate ESS.
   i. In the dialog box that appears, click the first hyperlink to log on to the **ESS** console.



   ii. Click **Activate Auto Scaling** to go to the **Enable Service** page.
   iii. Select the **I agree with Auto Scaling Agreement of Service** check box and click **Enable Now**.
   iv. On the **Activated** page, click **Console** to log on to the **ESS** console.
   v. Click **Go to Authorize** to go to the **Cloud Resource Access Authorization** page. Then, authorize ESS to access other cloud resources.
   vi. Click **Confirm Authorization Policy**.
   vii.

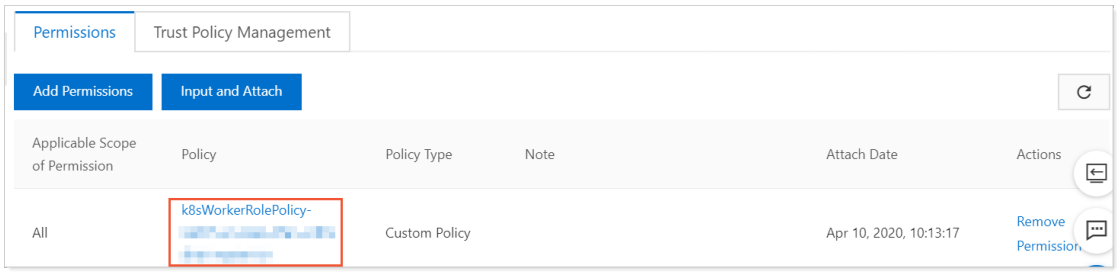
   If the authorization is successful, you are redirected to the ESS console. Close the page and modify the permissions of the worker RAM role.
2. Modify the permissions of the worker RAM role.

i. Click the second hyperlink in the dialog box to go to the **RAM Roles** page.

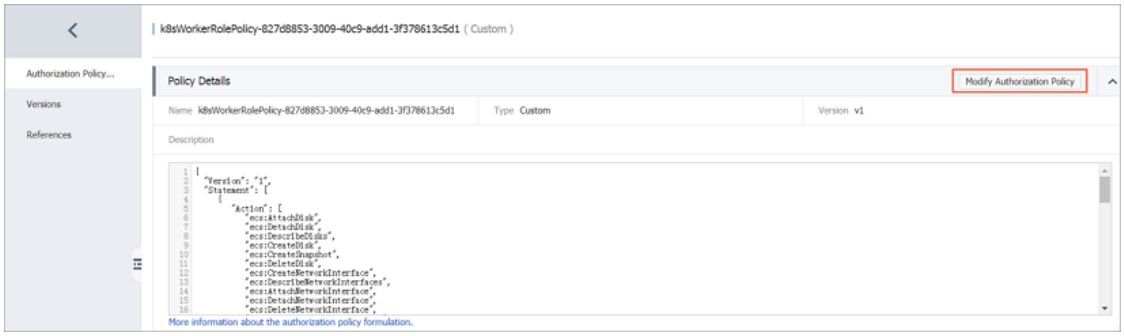


> **Note** You must log on to the RAM console by using an Alibaba Cloud account.

ii. On the **Permissions** tab, click the name of the policy assigned to the RAM role. The details page of the policy appears.



iii. Click **Modify Policy Document**. The **Modify Policy Document** panel appears on the right side of the page.

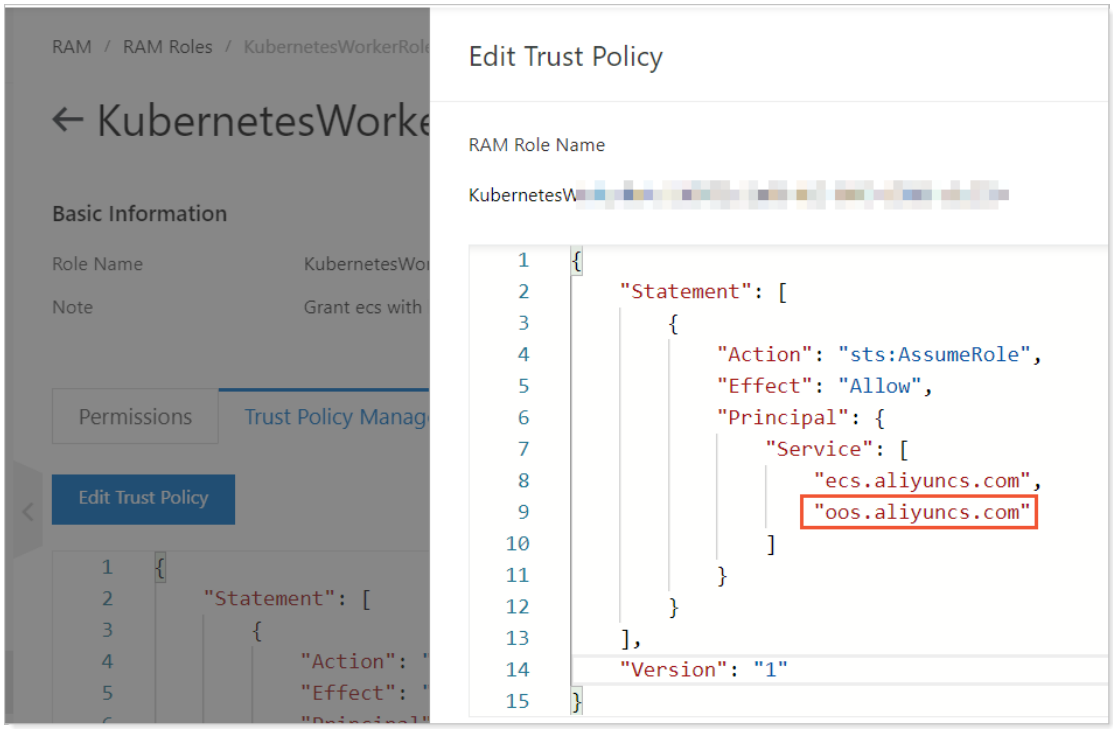


iv. In the **Policy Document** section, add the following policy content to the `Action` field and click **OK**.

```
"ecs:AllocateEipAddress",
"ecs:AssociateEipAddress",
"ecs:DescribeEipAddresses",
"ecs:DescribeInstanceTypes",
"ecs:DescribeInvocationResults",
"ecs:DescribeInvocations",
"ecs:ReleaseEipAddress",
"ecs:RunCommand",
"ecs:UnassociateEipAddress",
"ess:CompleteLifecycleAction",
"ess:CreateScalingRule",
"ess:DeleteScalingRule",
"ess:Describe*",
"ess:DetachInstances",
"ess:ExecuteScalingRule",
"ess:ModifyScalingGroup",
"ess:ModifyScalingRule",
"ess:RemoveInstances",
"vpc:AllocateEipAddress",
"vpc:AssociateEipAddress",
"vpc:DescribeEipAddresses",
"vpc:DescribeVSwitches",
"vpc:ReleaseEipAddress",
"vpc:UnassociateEipAddress",
"vpc:TagResources"
```

> **Note** Before you add the policy content, add a comma (,) to the end of the bottom line in the `Action` field.

v. On the **RAM Roles** page, click the name of the worker RAM role. On the details page of the RAM role, click the **Trust Policy Management** tab and click **Edit Trust Policy**. In the Edit Trust Policy panel, add oos.aliyuncs.com to the Service field, as shown in the following figure. Then, click **OK**.



ACK has limited permissions on nodes in the cluster

ACK has unlimited permissions on nodes in the cluster

An auto-scaling node pool in the cluster must be associated with an EIP

Step 3: Configure auto scaling

1. On the **Configure Auto Scaling** page, set the following parameters and click **Submit**.

Parameter	Description
Cluster	The name of the cluster for which you want to enable auto scaling.
Scale-in Threshold	For a scaling group that is managed by cluster-autoscaler, set the value to the ratio of the requested resources per node to the total resources per node. If the actual value is lower than the threshold, the node is removed from the cluster. **Note** In auto scaling, a scale-out event is automatically triggered based on node scheduling. Therefore, you need to set only scale-in parameters.

Parameter	Description
GPU Scale-in Threshold	The scale-in threshold for GPU-accelerated nodes. If the actual value is lower than the threshold, one or more GPU-accelerated nodes are removed from the Kubernetes cluster.
Defer Scale-in For	The amount of time that the cluster must wait before the cluster scales in. Unit: minutes. The default value is 10 minutes.
Cooldown	Newly added nodes cannot be removed in scale-in events during the cool down period.

2. Select an instance type. Supported instance types are regular instances, GPU-accelerated instances, and preemptible instances. Then, click **Create**.
3. In the **Auto Scaling Group Configuration** dialog box, set the following parameters to create a scaling group.

Parameter	Description
Region	The region where you want to deploy the scaling group. The scaling group and the Kubernetes cluster must be deployed in the same region. You cannot change the region after the scaling group is created.
VPC	The scaling group and the Kubernetes cluster must be deployed in the same VPC.
VSwitch	The vSwitches of the scaling group. You can specify vSwitches of different zones. The vSwitches allocate pod CIDR blocks to the scaling group.

4. Configure worker nodes.

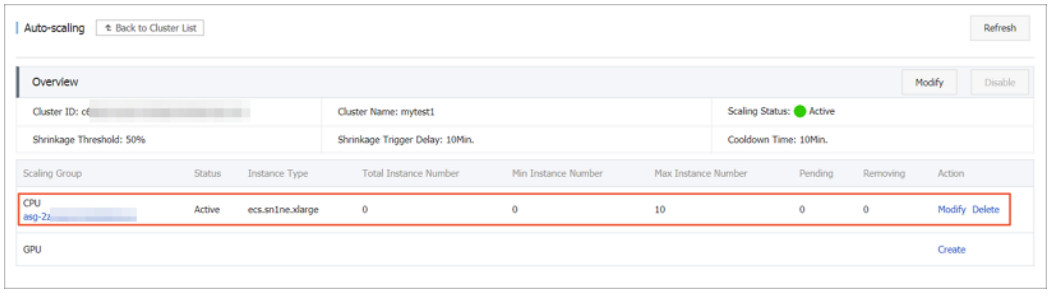
Parameter	Description
Node Type	The types of nodes in the scaling group. The node types must be the same as those selected when the cluster is created.
Instance Type	The instance types in the scaling group.
Selected Types	The instance types that you select. You can select at most 10 instance types.
System Disk	The system disk of the scaling group.
Mount Data Disk	Specify whether to mount data disks to the scaling group. By default, no data disk is mounted.

Parameter	Description
Instances	The number of instances contained in the scaling group. **Note** ○ Existing instances in the cluster are excluded. ○ By default, the minimum number of instances is 0. If you specify one or more instances, the system adds the instances to the scaling group. When a scale-out event is triggered, the instances in the scaling group are added to the cluster to which the scaling group is bound.
Key Pair	The key pair that is used to log on to the nodes in the scaling group. You can create key pairs in the ECS console. **Note** You can log on to the nodes only by using key pairs.
Scaling Mode	You can select **Standard** or **Swift**.
RDS Whitelist	The ApsaraDB RDS instances that can be accessed by the nodes added from the scaling group.
Node Label	Labels are automatically added to nodes that are added to the cluster by scale-out activities.
Taints	After you add taints to a node, ACK no longer schedules pods to the node.

5. Click **OK** to create the scaling group.

Check the results

1. On the **Auto Scaling** page, you can find the newly created scaling group below Regular Instance.



2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
4. On the **Deployments** tab, select the kube-system namespace. You can find the cluster-autoscaler component. This indicates that the scaling group is created.

FAQ

- Why does the auto scaling component fail to add nodes after a scale-out event is triggered?
  Check whether the following situations exist:
  - The instance types in the scaling group cannot fulfill the resource request from pods. By default, system components are installed for each node. Therefore, the requested pod resources must be less than the resource capacity of the instance type.
  - The worker RAM role does not have the required permissions. You must configure RAM roles for each Kubernetes cluster that is involved in the scale-out activity.
- Why does the auto scaling component fail to remove nodes after a scale-in event is triggered?
  Check whether the following situations exist:
  - The requested resource threshold of each pod is higher than the configured scale-in threshold.
  - Pods that belong to the *kube-system* namespace are running on the node.
  - A hard scheduling policy is configured to force the pods to run on the current node. Therefore, the pods cannot be scheduled to other nodes.
  - PodDisruptionBudget is set for the pods on the node and the minimum value of PodDisruptionBudget is reached.

  For more information about FAQ, see open source component.
- How does the system choose a scaling group for a scaling event?
  When pods cannot be scheduled to nodes, the auto scaling component simulates the scheduling of the pods based on the configurations of scaling groups. The configurations include labels, taints, and instance specifications. If a scaling group can simulate the scheduling of the pods, this scaling group is selected for the scale-out activity. If more than one scaling groups meet the requirements, the system selects the scaling group that has the fewest idle resources after simulation.

Related information

- Access the Kubernetes API server over the Internet
- HPA
- Implement horizontal auto scaling based on Alibaba Cloud metrics
- CronHPA
- Vertical pod autoscaling

13.3. HPA

You can create an application that has Horizontal Pod Autoscaling (HPA) enabled in the Container Service for Kubernetes (ACK) console. HPA can automatically scale container resources for your application. You can also use a YAML file to describe HPA settings.

Prerequisites

Before you enable HPA, make sure that you have completed the following steps:

- 创建Kubernetes托管版集群
- Connect to Kubernetes clusters by using kubectl

Create an application that has HPA enabled in the ACK console

ACK is integrated with HPA. You can create an application that has HPA enabled in the ACK console. You can enable HPA when you create an application or after the application is created.

Enable HPA when you create an application

1. Log on to the ACK console.

2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from Image**.
6. On the **Basic Information** wizard page, enter a name for your application, set other required parameters, and then click **Next**.

Parameter	Description
Name	The name of the application.
Replicas	The number of pods that are provisioned for the application. Default value: 2.
Type	The type of workload. You can select **Deployments**, **StatefulSets**, **Jobs**, **CronJobs**, or **DaemonSets**.
Label	Add a label to the application. The label is used to identify the application.
Annotations	Add an annotation to the application.
Synchronize Timezone	Specify whether to synchronize the time zone between nodes and containers.

7. On the **Container** wizard page, set the container parameters, select an image, and then configure the required computing resources. Click **Next**. For more information, see Configure containers for an application.

> **Note** You must configure the required computing resources for the Deployment. Otherwise, you cannot enable HPA.

8. On the **Advanced** wizard page, find the **Access Control** section, click **Create** on the right side of Services, and then set the parameters. For more information, see Configure advanced settings for an application.
9. On the **Advanced** wizard page, select **Enable** for **HPA** and configure the scaling threshold and related settings.
   - **Metric**: Select CPU Usage or Memory Usage. The selected resource type must be the same as the one that you have specified in the Required Resources field.
   - **Condition**: Specify the resource usage threshold. HPA triggers scaling activities when the threshold is exceeded.
   - **Max. Replicas**: Specify the maximum number of pods to which the Deployment can be scaled.
   - **Min. Replicas**: Specify the minimum number of pods that must run for the Deployment.
10. In the lower-right corner of the Advanced wizard page, click **Create**. The application is created with HPA enabled.
    Verify the result
    i. Click **View Details** or choose **Workloads > Deployments**. On the page that appears, click the **name of the created application** or click **Details** in the **Actions** column. Then, click the **Pod Scaling** tab to view information about the scaling group of the application.

ii. After the application starts to run, container resources are automatically scaled based on the CPU utilization. You can check whether HPA is enabled in the staging environment by performing a CPU stress test on the pods of the application. Verify that the pods are automatically scaled within 30 seconds.

Enable HPA after an application is created

This example describes how to enable HPA for a stateless application.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click the name of the application that you want to manage.
6. Click the **Pod Scaling** tab and click **Create**.
7. In the **Create** dialog box, configure the HPA settings. For more information about how to set the parameters, see HPA settings in Step 9.
8. Click **OK**.

Use kubectl to enable HPA

You can also create a Horizontal Pod Autoscaler by using an orchestration template and associate the Horizontal Pod Autoscaler with the Deployment for which you want to enable HPA. Then, you can use **kubectl** to enable HPA.

In the following example, HPA is enabled for an NGINX application.

1. Create a file named *nginx.yml* and copy the following content into the file.

   The following code block is a YAML template that is used to create a Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
    ports:
    - containerPort: 80
    resources:
     requests:                ##To enable HPA, you must set this parameter.
      cpu: 500m
```

2. Run the following command to create an NGINX application:

```
kubectl create -f nginx.yml
```

3. Create a Horizontal Pod Autoscaler.

   Use scaleTargetRef to associate the Horizontal Pod Autoscaler with the Deployment named **nginx**.

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
 name: nginx-hpa
 namespace: default
spec:
 scaleTargetRef:                ##Associate the Horizontal Pod Autoscaler with the nginx Deployment.
  apiVersion: apps/v1
  kind: Deployment
  name: nginx
 minReplicas: 1
 maxReplicas: 10
 metrics:
 - type: Resource
  resource:
   name: cpu
   targetAverageUtilization: 50
```

   > **Note** You must configure the requested resources for the pods of the application. Otherwise, the Horizontal Pod Autoscaler cannot be started.

4. Run the `kubectl describe hpa name` command. The following output is an example of a warning that is returned:

```
Warning  FailedGetResourceMetric    2m (x6 over 4m)  horizontal-pod-autoscaler  missing request for cpu on container nginx in pod default/nginx-deployment-basic-75675f5897-mqzs7
Warning  FailedComputeMetricsReplicas  2m (x6 over 4m)  horizontal-pod-autoscaler  failed to get cpu utilization: missing request for cpu on container nginx in pod default/nginx-deployment-basic-75675f5
```

5. After the Horizontal Pod Autoscaler is created, run the `kubectl describe hpa name` command.

   If the following output is returned, it indicates that the Horizontal Pod Autoscaler runs as expected:

```
Normal SuccessfulRescale 39s horizontal-pod-autoscaler New size: 1; reason: All metrics below target
```

   If the pod usage of the NGINX application exceeds 50% as specified in the HPA settings, the Horizontal Pod Autoscaler automatically adds pods. If the pod usage of the NGINX application drops below 50%, the Horizontal Pod Autoscaler automatically removes pods.

13.4. CronHPA

To avoid resource wasting in some scenarios, Container Service for Kubernetes (ACK) provides the kubernetes-cronhpa-controller component to automatically scale resources based on predefined schedules. This topic describes how to use Cron Horizontal Pod Autoscaler (CronHPA) to scale your workloads based on a schedule. This topic also describes how to enable CronHPA and Horizontal Pod Autoscaler (HPA) to interact without conflicts.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.

- Helm 2.11.0 or later is installed on your on-premises machine. For more information, see Install Helm.
- You are connected to the cluster by using kubectl before you run commands to perform the following operations. For more information, see Connect to Kubernetes clusters by using kubectl.

Context

kubernetes-cronhpa-controller is a Kubernetes HPA controller that can scale a Kubernetes cluster based on a schedule that is similar to a crontab. You can use CronHPA with Kubernetes objects whose subresources can be scaled. The objects include Deployments and StatefulSets. In addition, the subresources must be open source projects on GitHub. For more information, see kubernetes-cronhpa-controller.

The following table describes the parameters in the CronHPA configuration.

```
apiVersion: autoscaling.alibabacloud.com/v1beta1
kind: CronHorizontalPodAutoscaler
metadata:
 labels:
  controller-tools.k8s.io: "1.0"
 name: cronhpa-sample
 namespace: default
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment-basic
  excludeDates:
  # exclude November 15th
  - "* * * 15 11 *"
  # exclude every Friday
  - "* * * * * 5"
  jobs:
  - name: "scale-down"
    schedule: "30 */1 * * * *"
    targetSize: 1
  - name: "scale-up"
    schedule: "0 */1 * * * *"
    targetSize: 3
    runOnce: true
```

Parameter	Description
scaleTargetRef	scaleTargetRef specifies the object that you want to scale. If the subresources of the object can be scaled, you can enable CronHPA for the object.

Parameter	Description

excludeDates	The value of excludeDates must be an array of dates. Scaling jobs are not executed on dates that are specified in excludeDates.
	Note The minimum time period is **one day**.
	The value is in the `- "* * * * * *"` format, which indicates -"<seconds> <minutes> <hours> <days of month> <months>" format.
	For example, if you do not want to execute scaling jobs on November 15, set excludeDates to the following value:
	`excludeDates:`
	`- "* * * 15 11 *"`

jobs	You can set multiple CronHPA jobs in the spec section. You can set the following parameters for each CronHPA job:
	• name: Names are used to distinguish CronHPA jobs. Therefore, the name of each CronHPA job must be unique in the CronHPA configurations.
	• schedule: the scaling schedule, which is similar to a crontab. kubernetes-cronhpa-controller uses a Golang library to support a variety of rules. For more information, see go-cron. The format of cron expressions must conform to the following rules. Otherwise, you cannot create cron expressions.
	Create cron expressions based on the following rules:
	`Field name | Mandatory? | Allowed values | Allowed special characters`
	`---------- | ---------- | ---------- | ----------`
	`Seconds | Yes | 0-59 | * / , -`
	`Minutes | Yes | 0-59 | * / , -`
	`Hours | Yes | 0-23 | * / , -`
	`Day of month | Yes | 1-31 | * / , - ?`
	`Month | Yes | 1-12 or JAN-DEC | * / , -`
	`Day of week | Yes | 0-6 or SUN-SAT | * / , - ?`
	• targetSize: the number of pods to which you want to scale at the scheduled time.
	• runOnce: If you set runOnce to true, the job is executed only once. The job exits after one run.

Install the CronHPA controller

You can install the CronHPA controller ack-kubernetes-cronhpa-controller by using the following methods.

Method 1: Install the CronHPA controller on the Add-ons page in the ACK console

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. On the **Add-ons** page, click the **Others** tab, find **ack-kubernetes-cronhpa-controller**, and then click **Install**.

Method 2: Install the CronHPA controller from the Manage System Components menu

1. Log on to the ACK console.

2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the **Clusters** page, find the cluster that you want to manage and choose **More > Manage System Components** in the **Actions** column.
5. On the **Add-ons** page, click the **Others** tab, find **ack-kubernetes-cronhpa-controller**, and then click **Install**.

Method 3: Install the CronHPA controller from App Catalog

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. You can search for the component on the **Alibaba Cloud Apps** tab. In the upper-left corner of the page, enter *kubernetes-cronhpa-controller* into the **Name** search box and click the search icon. Find and click **ack-kubernetes-cronhpa-controller**.
4. On the right side of the **App Catalog - ack-kubernetes-cronhpa-controller** page, select a cluster to deploy the controller in the **Deploy** section and click **Create**.

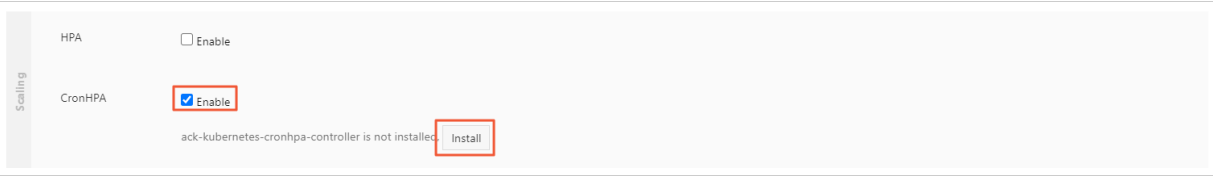
You can uninstall the CronHPA controller if CronHPA is no longer used. For more information about how to uninstall ack-kubernetes-cronhpa-controller, see Manage system components or Delete a release.

Create CronHPA jobs

Before you create and run CronHPA jobs for your application, make sure that the CronHPA controller runs as normal in your cluster and only one HPA task is created for your application. For more information about how to enable CronHPA and HPA to interact without conflicts, see Enable CronHPA and HPA to interact without conflicts. You can create CronHPA jobs in the following scenarios.

Scenario 1: Create CronHPA jobs when you create an application

In the **Scaling** section on the **Advanced** wizard page, select **Enable** on the right side of **CronHPA** to create CronHPA jobs for the application. For more information about how to create an application, see Create a stateless application by using a Deployment or Use a StatefulSet to create a stateful application.



The ACK console automatically checks whether the CronHPA controller is installed in the cluster. If the CronHPA controller is not installed, the **Install** button appears on the page. After the CronHPA controller is installed, the CronHPA parameters appear on the page. The following table describes the parameters.

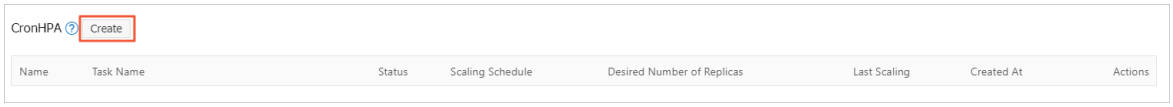
CronHPA parameters

Parameter	Description
Job Name	Enter a name for the CronHPA job. The name of each CronHPA job must be unique.
Desired Number of Replicas	Pod replicas are scaled to the desired number at the scheduled time.
Scaling Schedule	Set the scaling schedule. For more information about how to set the scaling schedule for a CronHPA job, see AliyunContainerService/kubernetes-cronhpa-controller.

Scenario 2: Create CronHPA jobs for an existing application

The following example demonstrates how to create CronHPA jobs for an existing application. A stateless application is used in this example.

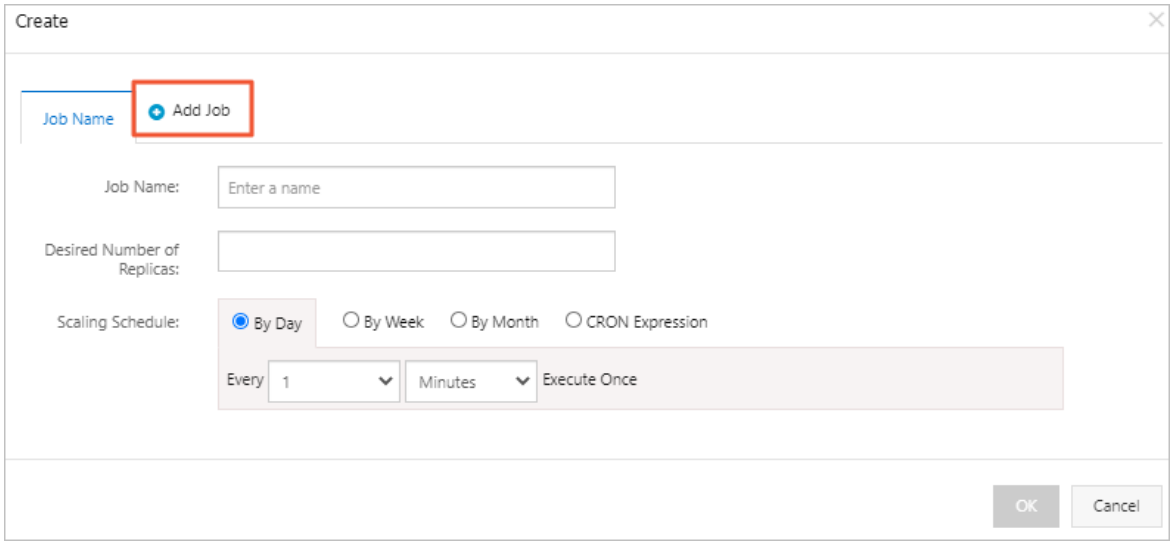
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, find the application that you want to manage and click **Details** in the **Actions** column.
6. Click the **Pod Scaling** tab and configure CronHPA jobs.
   - If the CronHPA controller is not installed, the **Install** button appears on the page. Click **Install** and perform the following steps.
   - If the CronHPA controller is installed, perform the following steps.
7. Click **Create** on the right side of **CronHPA**. In the **Create** dialog box, configure CronHPA jobs.



For more information about how to configure CronHPA jobs, see CronHPA parameters.

Create or modify CronHPA jobs

1. Go to the **Pod Scaling** tab by performing the steps described in the preceding Create CronHPA jobs for an existing application section.
2. On the **Pod Scaling** tab, find the created CronHPA job in the **CronHPA** section and click **Add or Modify Job** in the **Actions** column.
3. In the **Edit** dialog box, click **Add Job** to create CronHPA jobs. You can also modify existing CronHPA jobs. Click **OK**.



To delete CronHPA jobs, perform the following steps:

4. In the **Edit** dialog box, click the delete icon in the upper-right corner of the job that you want to delete. Click **OK**.

Enable CronHPA and HPA to interact without conflicts

CronHPA triggers horizontal scaling for containers based on schedules. HPA is used to scale pods to ensure the availability of your applications when network traffic spikes. If ACK detects that both CronHPA and HPA are deployed, ACK sets HPA as the scaling object of CronHPA. CronHPA triggers HPA to scale pods at the scheduled time.

The following YAML template shows the configurations of CronHPA and HPA:

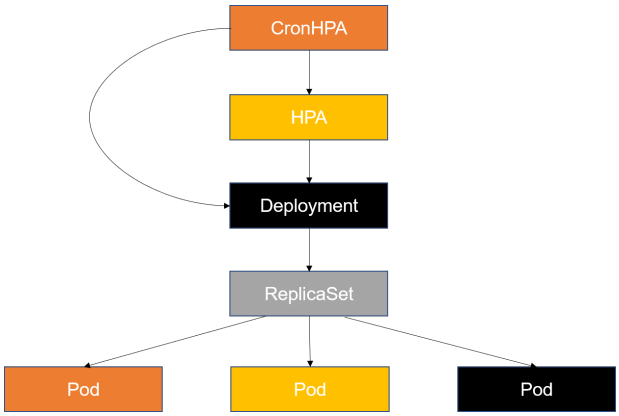
After you compare the configurations of CronHPA and HPA, you can find that:

- The scaleTargetRef field is used in the configurations of both CronHPA and HPA to specify the object to be scaled.
- The **crontab** rules in the jobs section of the CronHPA configuration specify the number to which pods are scaled at the scheduled time.
- HPA triggers scaling activities based on resource usage.

If both CronHPA and HPA are deployed, CronHPA and HPA may scale pods for the same application that is specified by scaleTargetRef. CronHPA and HPA are independent and unaware of each other. As a result, the CronHPA controller and the HPA controller scale pods for the application separately. The later scaling activity overwrites the earlier one.

Solution

The reason for the conflict between CronHPA and HPA is that the CronHPA controller and the HPA controller are unaware of each other. To resolve the conflict, CronHPA needs only to detect the state of HPA. In the HPA configuration, scaleTargetRef specifies the Deployment that is managed by HPA. The Deployment is related to a ReplicaSet. When HPA scales the Deployment, the ReplicaSet scales pods to the desired number. ACK modifies the CronHPA configuration by setting scaleTargetRef to HPA. CronHPA can find the application that is specified by scaleTargetRef in the HPA configuration. This enables CronHPA to detect the state of HPA.



The following YAML template shows the configurations that enable CronHPA and HPA to interact without conflicts:

```
apiVersion: autoscaling.alibabacloud.com/v1beta1
kind: CronHorizontalPodAutoscaler
metadata:
 labels:
  controller-tools.k8s.io: "1.0"
 name: cronhpa-sample
spec:
  scaleTargetRef:
    apiVersion: autoscaling/v1
    kind: HorizontalPodAutoscaler
    name:  nginx-deployment-basic-hpa
  jobs:
  - name: "scale-down"
    schedule: "30 */1 * * * *"
    targetSize: 1
    runOnce: true
  - name: "scale-up"
    schedule: "0 */1 * * * *"
    targetSize: 3
    runOnce: true
```

After you deploy the preceding YAML template, CronHPA is aware of the values of minReplicas, maxReplicas, and desiredReplicas in the HPA configuration. CronHPA is also aware of the current number of pods provisioned for the application that is specified by scaleTargetRef. CronHPA can detect the state of HPA by modifying the HPA configurations. CronHPA compares the desired number of pods with the current number of pods and then determines whether to trigger scaling activities and change the maximum number of pods in the HPA configuration. CronHPA also compares the desired number of pods with the maximum and minimum numbers of pods that are specified in the HPA configuration and then determines whether to change the minimum numbers of pods in the HPA configuration.

The following table describes the rules that enable CronHPA and HPA to interact without conflicts.

HPA (min/max)	Cronhpa	Deployment	Scaling result	Description
1/10	5	5	• HPA (min/max): 1/10 • Deployment: 5	If the number of pods desired by CronHPA equals the current number of pods, CronHPA does not change the maximum and minimum numbers of pods in the HPA configuration. In addition, no scaling activity is triggered.
1/10	4	5	• HPA (min/max): 1/10 • Deployment: 5	If the number of pods desired by CronHPA is smaller than the current number of pods, no scaling activity is triggered.

HPA (min/max)	Cronhpa	Deployment	Scaling result	Description
1/10	6	5	• HPA (min/max): 6/10 • Deployment: 6	• If the number of pods desired by CronHPA is greater than the current number of pods, CronHPA adds pods to reach the desired number. • If the number of pods desired by CronHPA is greater than the value of minReplicas in the HPA configuration, CronHPA changes the value of minReplicas.
5/10	4	5	• HPA (min/max): 4/10 • Deployment: 5	• If the number of pods desired by CronHPA is smaller than the current number of pods, no scaling activity is triggered. • If the number of pods desired by CronHPA is smaller than the value of minReplicas in the HPA configuration, CronHPA changes the value of minReplicas.
5/10	11	5	• HPA (min/max): 11/11 • Deployment: 11	• If the number of pods desired by CronHPA is greater than the current number of pods, CronHPA adds pods to reach the desired number. • If the number of pods desired by CronHPA is greater than the value of maxReplicas in the HPA configuration, CronHPA changes the value of maxReplicas.

The following list describes the parameters in the table:

- HPA (min/max): the minimum and maximum numbers of pods that are specified in the HPA configuration.
- CronHPA: the desired number of pods.
- Deployment: the current number of pods that are provisioned for the application.

CronHPA does not directly change the number of pods for the Deployment. It triggers HPA to scale the pods. This resolves the conflict between CronHPA and HPA.

CronHPA HPA

Related information

- HPA

13.5. Vertical pod autoscaling

You can deploy the vertical-pod-autoscaler component to a Container Service for Kubernetes (ACK) cluster. The Vertical Pod Autoscaler (VPA) enables vertical autoscaling of pods. The VPA automatically sets limits on the resource usage of a cluster based on the pod resource usage. In this case, ACK can schedule the pods to nodes that have sufficient resources. The VPA also maintains the ratio of requests to limits that you specify in the initial container configuration. This topic describes how to use a YAML file to enable vertical pod autoscaling.

Prerequisites

Make sure that the following operations are performed:

- An ACK cluster is created and its version is later than 1.12. For more information, see 创建Kubernetes托管版集群.
- The cluster is connected by using a command-line tool. For more information, see Connect to Kubernetes clusters by using kubectl.
- The VPA deployed to the cluster is uninstalled. Otherwise, the newly created VPA may conflict with the original VPA.

Context

> **Notice** Vertical pod autoscaling is in testing. Use this feature with caution.

- You can use the VPA to update the resource configurations of the running pods. This feature is in testing. The configuration updates will lead to pod restart and rebuilding, and the pods may be scheduled to other nodes.
- The VPA does not evict the pods that are not managed by a replication controller. For these pods, the Auto mode is equivalent to the Initial mode.
- The VPA cannot run in conjunction with the Horizontal Pod Autoscaler (HPA) that monitors the CPU and memory metrics. If the HPA monitors only other custom or external resource metrics, you can use the VPA in conjunction with the HPA.
- The VPA uses an admission webhook as its admission controller. If other admission webhooks exist in the cluster, make sure that the admission webhooks do not conflict with the admission webhook of the VPA. The execution sequence of admission controllers is defined in the configuration parameters of the API server.
- The VPA can react to most out-of-memory (OOM) events, but may fail to handle some events in some scenarios.
- The VPA performance is not tested in large-scale clusters.
- The pod resource requests set by the VPA may exceed the upper limit of the actual resources, including node resources, idle resources, and resource quotas. In this case, a pod may enter the Pending state and cannot be scheduled. You can use the cluster autoscaler to mitigate this issue.
- If multiple VPAs monitor the resource usage of a pod at the same time, some undefined behavior may occur.

Install vertical-pod-autoscaler

1. Run the following command to create a custom resource definition (CRD) for vertical-pod-autoscaler.

   The CRD improves the scalability of ACK clusters. For more information, see Extend the Kubernetes API with CustomResourceDefinitions.

```
kubectl apply -f crd.yaml
```

Add the following content to the *crd.yaml* file:

```
---
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
 name: verticalpodautoscalers.autoscaling.k8s.io
spec:
 group: autoscaling.k8s.io
 scope: Namespaced
 names:
  plural: verticalpodautoscalers
  singular: verticalpodautoscaler
  kind: VerticalPodAutoscaler
  shortNames:
   - vpa
 version: v1beta1
 versions:
  - name: v1beta1
    served: true
    storage: false
  - name: v1beta2
    served: true
    storage: true
 validation:
  # openAPIV3Schema is the schema for validating custom objects.
  openAPIV3Schema:
    type: object
    properties:
      spec:
       type: object
       required: []
       properties:
        targetRef:
          type: object
        updatePolicy:
          type: object
          properties:
            updateMode:
              type: string
        resourcePolicy:
          type: object
          properties:
            containerPolicies:
              type: array
              items:
                type: object
---
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
 name: verticalpodautoscalercheckpoints.autoscaling.k8s.io
spec:
 group: autoscaling.k8s.io
 scope: Namespaced
```

```
  names:
    plural: verticalpodautoscalercheckpoints
    singular: verticalpodautoscalercheckpoint
    kind: VerticalPodAutoscalerCheckpoint
    shortNames:
      - vpacheckpoint
  version: v1beta1
  versions:
    - name: v1beta1
      served: true
      storage: false
    - name: v1beta2
      served: true
      storage: true
```

2. Install the components of vertical-pod-autoscaler.

   vertical-pod-autoscaler contains the following components: admission-controller, recommender, and updater.

   Install admission-controller | Install recommender | Install updater

Verify that the VPA is installed

1. Use the following YAML file to define and create a deployment named nginx-deployment-basic and a VPA resource named nginx-deployment-basic-vpa.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment-basic
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
---
apiVersion: autoscaling.k8s.io/v1beta1
kind: VerticalPodAutoscaler
metadata:
 name: nginx-deployment-basic-vpa
spec:
 targetRef:
  apiVersion: "apps/v1"
  kind:    Deployment
  name:    nginx-deployment-basic
 updatePolicy:
  updateMode: "Off"
```

> **Note** Set updateMode to Off, and leave the requests and limits fields in the deployment unspecified.

2. Run the following command to query the CPU and memory resource requests recommended by the VPA for the deployment:

```
kubectl describe vpa nginx-deployment-basic-vpa
```

The following command output shows the recommended resource requests:

```
recommendation:
  containerRecommendations:
  - containerName: nginx
    lowerBound:
      cpu: 50m
      memory: 300144k
    target:
      cpu: 50m
      memory: 300144k
    upperBound:
      cpu: 8031m
      memory: 800000k
```

You can set the resource requests for the deployment based on the recommendation. The VPA performs continuous monitoring of the resource usage of the deployment and provides optimization suggestions.

Related information

- HPA

13.6. Implement horizontal auto scaling based on Alibaba Cloud metrics

This topic describes how to implement auto scaling based on Alibaba Cloud metrics.

Prerequisites

创建Kubernetes托管版集群

Context

In many scenarios, you may want to scale the number of pods based on metrics such as the HTTP request rate and the queries per second (QPS) of the Ingress. By default, HPA does not support custom or external metrics. However, Kubernetes provides the external metrics mechanism that allows you to implement auto scaling in a flexible manner.

Deploy alibaba-cloud-metrics-adapter

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab. Select **Auto Scaling** and click **ack-alibaba-cloud-metrics-adapter**.
4. In the **Deploy** section, select **Cluster** and click **Create**.
   In the left-side navigation pane of the ACK console, click **Clusters**. On the **Clusters** page, find the cluster in which the component is deployed and click **Details** in the **Actions** column. On the details page of the cluster, click **Releases**. On the **Helm** page, you can view that **ack-alibaba-cloud-metrics-adapter** is deployed in the cluster.

Example

The following example shows how to configure HPA by creating a Deployment and a Service named nginx.

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name or click **Details** in

the **Actions** column.

4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Set **Sample Template** to **Custom**. Use the following YAML template to create a Deployment and a ClusterIP type Service. Then, click **Create**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment-basic
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
 name: nginx
 namespace: default
spec:
 ports:
  - port: 80
   protocol: TCP
   targetPort: 80
 selector:
  app: nginx
 type: ClusterIP
```

7. In the left-side navigation pane, click **Services and Ingresses > Ingresses**. On the **Ingresses** page, click **Create** in the upper-right corner.
8. In the **Create** dialog box, set the parameters and click **Create**. After you create an Ingress, the **Ingresses** page appears.
9. On the Ingresses page, find the newly created Ingress and click **Details** in the Actions column to view information about the Ingress.
10. Configure HPA.
    i. In the left-side navigation pane of the ACK console, choose **Marketplace > Orchestration Templates**.
    ii. On the **Templates** page, select **hpa** and click **Create Application** to deploy an application with the following YAML file:

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
 name: ingress-hpa
spec:
 scaleTargetRef:
  apiVersion: apps/v1
  kind: Deployment
  name: nginx-deployment-basic
 minReplicas: 2
 maxReplicas: 10
 metrics:
  - type: External
   external:
    metric:
     name: sls_ingress_qps
     selector:
      matchLabels:
       sls.project: "***" # Replace sls.project with the actual value.
       sls.logstore: "nginx-ingress"
       sls.ingress.route: "default-nginx-80"
    target:
     type: AverageValue
     averageValue: 10
  - type: External
   external:
    metric:
     name: sls_ingress_latency_p9999
     selector:
      matchLabels:
       # default ingress log project is k8s-log-clusterId
       sls.project: "***"
       # default ingress logstre is nginx-ingress
       sls.logstore: "nginx-ingress"
       # namespace-svc-port
       sls.ingress.route: "default-nginx-80"
       # sls vpc endpoint, default true
       # sls.internal.endpoint:ture
    target:
     type: Value
     # sls_ingress_latency_p9999 > 10ms
     value: 10
```

The following table describes the parameters that are used to configure HPA.

Parameter	Description

Parameter	Description
sls.ingress.route	Set the value in the <namespace>-<svc>-<port> format. For example, default-nginx-80. This parameter is required. Note <namespace> specifies the namespace to which the Ingress belongs. <svc> specifies the name of the Service that you selected when you created the Ingress. <port> specifies the port of the Service.
sls.logstore	The name of the Logstore in Log Service. This parameter is required.
sls.project	The name of the Log Service project. This parameter is required.
sls.internal.endpoint	Specifies whether to access Log Service over the internal network or the Internet. Default value: true. If you set the value to true, you access Log Service over the internal network. If you set the value to false, you access Log Service over the Internet.

Note
The sls_ingress_qps and sls_ingress_latency_p9999 metrics are used by HPA to automatically scale the number of pods. In the target sections, each metric has a different type value:

- The type value of the sls_ingress_qps metric is set to AverageValue. This indicates that the metric value is the result of dividing the total QPS by the number of pods.
- The type value of the sls_ingress_latency_p9999 metric is set to Value. This indicates that the latency is not divided by the number of pods.

The two type values are commonly used in HPA configurations.

iii. In the upper-right corner of the **Templates - HPA** page, click **Create**.

11. After HPA is configured, run the following script to perform a stress test:

```
#!/bin/bash
##Use Apache Benchmark to send requests to the Service exposed by the Ingress. The test lasts 300 seconds and 10 concurrent requests are sent per second.
ab -t 300 -c 10 <The domain name of the Ingress>
```

12. Check whether HPA works as expected.

i. In the left-side navigation pane of the ACK console, click **Clusters**. On the **Clusters** page, find the cluster that you want to manage and choose **More > Open Cloud Shell** in the **Actions** column.

ii. Run the `kubectl get hpa ingress-hpa` command to check the number of pods after the scale-out. If the value of REPLICAS is the same as the value of MAXPODS, it indicates that HPA scaled out the number of pods as expected.

```
$ kubectl get hpa ingress-hpa
NAME          REFERENCE                                TARGETS       MINPODS   MAXPODS   REPLICAS   AGE
ingress-hpa   Deployment/nginx-deployment-basic   27/10 (avg)   2         10        10         7m49s
```

FAQ

- What do I do if the TARGETS column shows <unknow> after I run the `kubectl get hpa` command? Perform the following operations to troubleshoot the issue:
  i. Run the `kubectl describe hpa <hpa_name>` command to check why HPA does not function as normal.
     - If the value of AbleToScale is False in the Conditions field, check whether the Deployment is created as normal.
     - If the value of ScalingActive is False in the Conditions field, proceed to the next step.
  ii. Run the `kubectl get --raw "/apis/external.metrics.k8s.io/v1beta1/"` command. If `Error from server (NotFound): the server could not find the requested resource` is returned, verify the status of alibaba-cloud-metrics-adapter.
  If the status of alibaba-cloud-metrics-adapter is normal, check whether the HPA metrics are relevant to the Ingress. If the metrics are relevant to the Ingress, make sure that you deploy the Log Service component before ack-alibaba-cloud-metrics-adapter is deployed. For more information, see Monitor nginx-ingress and analyze the access log of nginx-ingress.
  iii. Make sure that the values of the HPA metrics are valid. The value of sls.ingress.route must be in the `<namespace>-<svc>-<port>` format.
     - `<namespace>` specifies the namespace to which the Ingress belongs.
     - `<svc>` specifies the name of the Service that you selected when you created the Ingress.
     - `<port>` specifies the port of the Service.
- Where can I find the metrics that are supported by HPA?
  For more information about the metrics that are supported by HPA, see Alibaba Cloud metrics adapter.
  The following table describes the commonly used metrics.

Metric	Description	Additional parameter
sls_ingress_qps	The number of requests that the Ingress can process per second based on a specific routing rule.	sls.ingress.route
sls_ingress_latency_avg	The average latency of all requests.	sls.ingress.route
sls_ingress_latency_p50	The maximum latency for the fastest 50% of all requests.	sls.ingress.route
sls_ingress_latency_p95	The maximum latency for the fastest 95% of all requests.	sls.ingress.route
sls_ingress_latency_p99	The maximum latency for the fastest 99% of all requests.	sls.ingress.route
sls_ingress_latency_p9999	The maximum latency for the fastest 99.99% of all requests.	sls.ingress.route

Metric	Description	Additional parameter
sls_ingress_inflow	The inbound bandwidth of the Ingress.	sls.ingress.route

13.7. Use ECI elastic scheduling

Elastic Container Instance (ECI) elastic scheduling is an elastic scheduling strategy provided by Alibaba Cloud. You can add annotations to specify the resources that you want to use when you deploy applications. You can use only Elastic Compute Service (ECS) instances or elastic container instances, or automatically request elastic container instances when ECS resources are insufficient. ECI elastic scheduling can meet your resource requirements in different workload scenarios.

For more information, see Use ECI elastic scheduling.

13.8. ACK KEDA

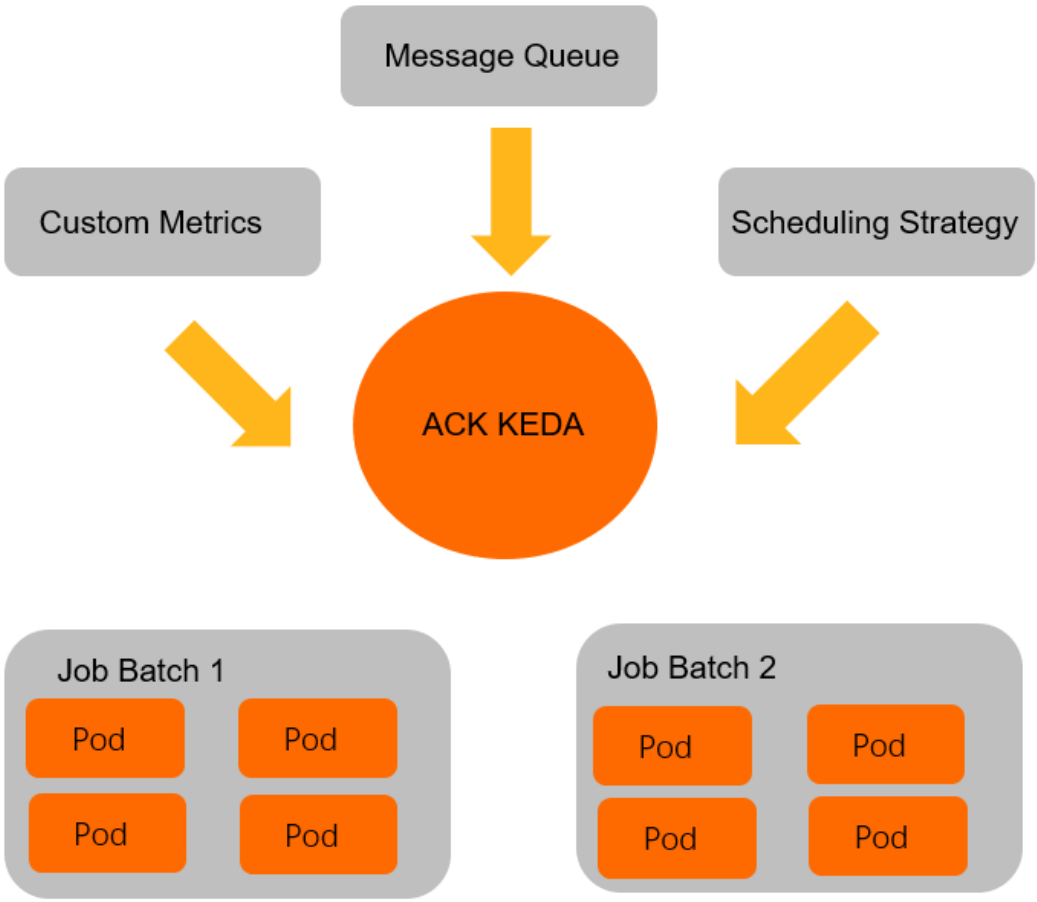
Container Service for Kubernetes (ACK) supports Kubernetes-based Event Driven Autoscaling (KEDA). You can install KEDA in the ACK console to implement event-driven scaling for clusters. This topic describes what KEDA is, how KEDA works, and how to use KEDA.

Overview

For Kubernetes, Horizontal Pod Autoscaler (HPA) is the most commonly used solution to automatically scale pods. HPA determines scaling strategies based on the differences between the resource usage and predefined thresholds. HPA is an easy-to-use tool that supports a wide array of resource metrics. However, it does not support real-time scaling. For example, HPA cannot scale resources when certain events are detected. To address this issue, you can install KEDA in the ACK console. ACK KEDA supports event-driven scaling that can be used in various event-driven scenarios, such as video and audio transcoding on demands, event-driven jobs, and stream processing.

How KEDA works

ACK KEDA is an enhanced version of the open source KEDA. It supports event-driven scaling. The following figure shows how ACK KEDA works.



ACK KEDA periodically consumes data from event sources. When pending messages increase, ACK KEDA is triggered to scale a batch of jobs. After the next period starts, the next batch of jobs is asynchronously scaled. ACK KEDA supports the following features:

- Supports a wide array of event sources
  ACK KEDA supports data sources such as Kafka, MySQL, PostgreSQL, RabbitMQ, and MongoDB. For more information, see RabbitMQ Queue.
- Controls the concurrency of jobs
  When a large number of jobs are submitted, the stability of the underlying control system is adversely affected because the system must holistically control resources, quotas, and API requests. ACK KEDA can control the concurrency of jobs in one or more batches to ensure system stability.
- Clears metadata after jobs are completed
  A large amount of metadata is retained after a great number of jobs are completed. The increase of metadata degrades the stability of the API server. The performance and stability of the cluster are also degraded and other services may be adversely affected. ACK KEDA can automatically clear metadata after jobs are completed.

Case study

A simple transcoding job is used in the following case. When a new job is received, a piece of data similar to the following example is inserted to MongoDB: `{"type":"mp4","state":"waiting","createTimeStamp":"1610332940","fileName":"World and peace","endTimeStamp":"","uuid":"1fae72ff-3239-42f5-af97-04711d8007e8"}`. ACK KEDA queries the database and detects data entries that meet the `"state":"waiting"` condition. Then, ACK KEDA creates pods to process the data entries. One pod is created for one data entry. After the transcoding is completed, the value of the `state` field is changed from `waiting` to `finished`. After the job is completed, the metadata is automatically cleared to reduce the load on the API Server. This allows you to manage jobs in a convenient manner.

Step 1: Deploy ACK KEDA

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, enter ack-keda in the search box in the upper-right corner to search for ACK KEDA. Then, click the ack-keda card.
4. On the **App Catalog - ack-keda** page, select the cluster where you want to deploy ACK KEDA in the **Deploy** section, and click **Create**.
   In the left-side navigation pane, click **Clusters**. On the **Clusters** page, select the cluster where ACK KEDA is deployed and click the cluster name or click **Details** in the **Actions** column. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**. In the upper-left corner of the **Deployments** page, set **Namespace** to kube-system and verify that ACK KEDA is displayed.

Step 2: Add MongoDB as an event source

1. Deploy MongoDB.

   If you have already deployed MongoDB, skip this step.

   > **Notice** The database is used only for testing purposes. Do not use the database in production environments.

   i. Create a YAML file named *mongoDB.yaml* and add the following code to the file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: mongodb
spec:
 replicas: 1
 selector:
  matchLabels:
   name: mongodb
 template:
  metadata:
   labels:
    name: mongodb
  spec:
   containers:
   - name: mongodb
    image: mongo:4.2.1
    imagePullPolicy: IfNotPresent
    ports:
    - containerPort: 27017
     name: mongodb
     protocol: TCP
---
kind: Service
apiVersion: v1
metadata:
 name: mongodb-svc
spec:
 type: ClusterIP
 ports:
 - name: mongodb
  port: 27017
  targetPort: 27017
  protocol: TCP
 selector:
  name: mongodb
```

ii. Deploy MongoDB to the mongodb namespace of the cluster.

```
kubectl apply -f mongoDB.yaml -n mongodb
```

2. Log on to MongoDB and create a user account.

i. Add a user account.

```
//Create a user account
kubectl exec -n mongodb mongodb-xxxxx -- mongo --eval 'db.createUser({ user:"test_user",pwd:"test_p
assword",roles:[{ role:"readWrite", db: "test"}]})'
```

ii. Log on to MongoDB.

```
//Logon authentication.
kubectl exec -n mongodb mongodb-xxxxx -- mongo --eval  'db.auth("test_user","test_password")'
```

iii. Create a collection.

```
// Create a collection
kubectl exec -n mongodb mongodb-xxxxx -- mongo --eval  'db.createCollection("test_collection")'
```

3. Deploy TriggerAuthentication and a Secret.

   ACK KEDA uses TriggerAuthentication to authenticate logon requests to event sources. In this case, MongoDB is added as an event source. ACK KEDA uses the `secretTargetRef` field in TriggerAuthentication to retrieve the connection information from the specified Secret and then uses the connection information to authenticate requests to MongDB.

   i. Create a YAML file named *auth.yaml* and add the following code to the file:

   ```
   apiVersion: keda.sh/v1alpha1
   kind: TriggerAuthentication
   metadata:
    name: mongodb-trigger
   spec:
    secretTargetRef:
      - parameter: connectionString
        name: mongodb-secret
        key: connect
   ---
   apiVersion: v1
   kind: Secret
   metadata:
    name: mongodb-secret
   type: Opaque
   data:
    connect: bW9uZ29kYjovL3Rlc3RfdXNlcjp0ZXN0X3Bhc3N3b3JkQG1vbmdvZGItc3ZjLm1vbmdvZGIuc3ZjLmNsdXN0ZXIubG9jYWw6MjcwMTcvdGVzdA==
   ```

   ii. Deploy TriggerAuthentication to the mongodb-test namespace of the cluster.

   ```
   kubectl apply -f auth.yaml -n mongodb-test
   ```

4. Deploy ScaledJob.

   ScaledJob is used to define the job template and specify the database to be queried and the query expression. In the following example, a job that queries the test collection collection in the test database and transcodes data entries that meet the `{"type":"mp4","state":"waiting"}` condition is created.

i. Create a YAML file named *job.yaml* and add the following code to the file:

```
apiVersion: keda.sh/v1alpha1
kind: ScaledJob
metadata:
 name: mongodb-job
spec:
 jobTargetRef:
  // Job template.
  template:
   spec:
    containers:
     - name: mongo-update
       image: registry.cn-hangzhou.aliyuncs.com/carsnow/mongo-update:v6
       args:
        - --connectStr=mongodb://test_user:test_password@mongodb-svc.mongodb.svc.cluster.local:27
017/test
        - --dataBase=test
        - --collection=test_collection
       imagePullPolicy: IfNotPresent
    restartPolicy: Never
  backoffLimit: 1
 pollingInterval: 15
 maxReplicaCount: 5
 successfulJobsHistoryLimit: 0
 failedJobsHistoryLimit: 10
 triggers:
  - type: mongodb
    metadata:
     dbName: test                    //The database to be queried.
     collection: test_collection          //The collection to be queried.
     query: '{"type":"mp4","state":"waiting"}'  //Create a job to process each data entry whose type is set
to mp4 and state is set to waiting.
     queryValue: "1"
    authenticationRef:
     name: mongodb-trigger
```

`query` : Set a condition. When ACK KEDA detects data entries that meet the specified condition, jobs are started.

ii. Deploy ScaledJob to the mongodb-test namespace of the cluster.

```
kubectl apply -f scaledJob.yaml -n mongodb-test
```

5. Insert five data entries to MongoDB.

```
//Insert five data entries to MongoDB.
kubectl exec -n mongodb mongodb-xxxxx -- mongo --eval 'db.test_collection.insert([
 {"type":"mp4","state":"waiting","createTimeStamp":"1610352740","fileName":"My Love","endTimeStamp":"","uuid":"1gae72ff-3239-42f5-af97-04711d8007e8"},
 {"type":"mp4","state":"waiting","createTimeStamp":"1610350740","fileName":"Harker","endTimeStamp":"","uuid":"1gae72ff-3239-42f5-af97-04711d8007e8"},
 {"type":"mp4","state":"waiting","createTimeStamp":"1610152940","fileName":"The World","endTimeStamp":"","uuid":"1gae72ff-3239-42f5-af97-04711d87767e8"},
 {"type":"mp4","state":"waiting","createTimeStamp":"1610390740","fileName":"Mother","endTimeStamp":"","uuid":"1gae72ff-3239-42f5-af97-04799d8007e8"},
 {"type":"mp4","state":"waiting","createTimeStamp":"1610344740","fileName":"Jagger","endTimeStamp":"","uuid":"1gae72ff-3239-42f5-af97-04711d80099e8"},
 ])'
```

Step 3: Check whether ACK KEDA works as expected

Run the following command to query jobs:

```
//watch job
watch -n 1 kubectl get job -n mongodb-test
```

```
Every 1.0s: kubectl get job -n mongodb-test                    iZt4nbpmuvh2hxnjuqdmr4Z: Sun Jan 10 22:09:45 2021

NAME                COMPLETIONS   DURATION   AGE
mongodb-job-2qv7m   1/1           1s         3s
mongodb-job-f2ccp   0/1           3s         3s
mongodb-job-fbfs9   1/1           1s         3s
mongodb-job-kjrmd   0/1           3s         3s
mongodb-job-p48w7   0/1           3s         3s
```

Verify that five jobs are created. Log on to MongoDB and check the inserted data. Verify that the state of each data entry that you inserted is changed from `waiting` to `finished`.

```
  db.test_collection.find()
{ "_id" : ObjectId("5ffd37a95217f9cfe[redacted]") }
{ "_id" : ObjectId("5ffd38105217f9cfe[redacted]"), "type" : "mp4", "state" : "finished", "createTimeStamp" : "1610352740", "fileName" : "My Love", "endTimeStamp" : "", "uuid" : "1gae72ff-3239-42f5-af97-04711[redacted]" }
{ "_id" : ObjectId("5ffd38335217f9cfe[redacted]"), "type" : "mp4", "state" : "finished", "createTimeStamp" : "1610350740", "fileName" : "Harker", "endTimeStamp" : "", "uuid" : "1gae72ff-3239-42f5-af97-04711[redacted]" }
{ "_id" : ObjectId("5ffd38585217f9cfe[redacted]"), "type" : "mp4", "state" : "finished", "createTimeStamp" : "1610152940", "fileName" : "The World", "endTimeStamp" : "", "uuid" : "1gae72ff-3239-42f5-af97-04711d8[redacted]" }
{ "_id" : ObjectId("5ffd38735217f9cfe[redacted]"), "type" : "mp4", "state" : "finished", "createTimeStamp" : "1610390740", "fileName" : "Mother", "endTimeStamp" : "", "uuid" : "1gae72ff-3239-42f5-af97-04799d8[redacted]" }
{ "_id" : ObjectId("5ffd388a5217f9cfe7[redacted]"), "type" : "mp4", "state" : "finished", "createTimeStamp" : "1610344740", "fileName" : "Jagger", "endTimeStamp" : "", "uuid" : "1gae72ff-3239-42f5-af97-04711d8[redacted]" }
>
```

13.9. FAQ about HPA

Horizontal Pod Autoscaler (HPA) is a component that can automatically scale the number of pods in Kubernetes clusters. This topic provides answers to some frequently asked questions about HPA.

- Issue 1: What do I do if a FailedGetResourceMetric warning is returned for HPA? ❯
- Issue 2: What do I do if excess pods are added by HPA during a rolling update? ❯
- Issue 3:What do I do if HPA does not scale pods when the scaling threshold is reached? ❯
- Issue 4: How do I set the data collection interval for HPA? ❯
- Issue 5: Can CronHPA and HPA interact without conflicts? ❯
- [redacted]: the issue that excess pods are added by HPA when CPU or memory usage rapidly increases? ❯

Related information

- metrics-server
- HPA

- CronHPA
- Install the metrics-server component

14.Release management

14.1. Use Ingresses to implement canary releases

When you upgrade your application versions, you can implement rolling updates, phased releases, blue-green releases, and canary releases. This topic describes how to implement canary releases for applications in a Container Service for Kubernetes (ACK) cluster by using Ingress controllers.

Prerequisites

创建Kubernetes托管版集群

Context

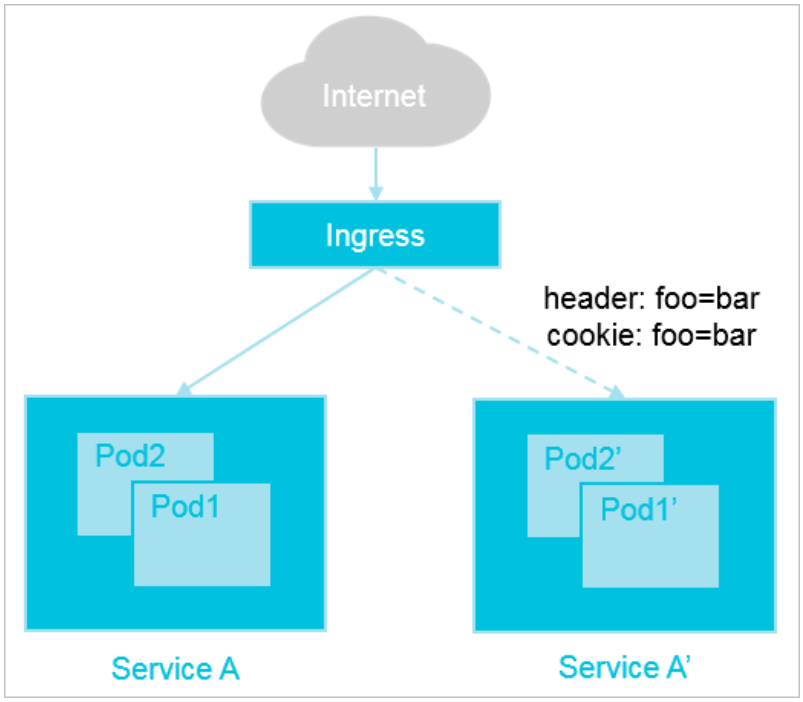
You can implement a canary release or a blue-green release to publish two identical production environments for an earlier application version and a new application version. In this case, when users send requests, ACK routes some requests to the new version based on specific rules. If the new version runs as normal for a period of time, you can switch all traffic from the earlier version to the new version.

A/B testing is a way of implementing canary releases. In A/B testing, some users use the earlier version, and requests from the other users are forwarded to the new version. If the new version runs as normal for a period of time, you can gradually switch all traffic to the new version.

Scenarios

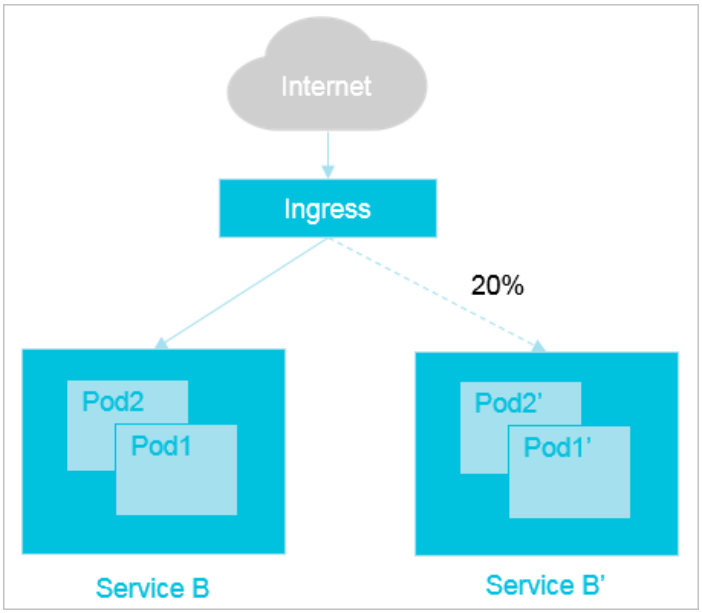
Traffic splitting based on requests

Assume that you have already created Service A for your production environment to provide Layer 7 access for users. When new features are available, you need to create Service A' for the new application version. If you want to keep Service A for external access, you can forward requests whose values of the foo parameters in the request headers match `bar` or whose values of the foo parameters in the cookie match `bar` to Service A'. If the new version stably runs for a period of time, you can switch all traffic from Service A to Service A'. Then, you can delete Service A.



Traffic splitting based on Service weights

Assume that you have already created Service B in your production environment to provide Layer 7 access for users. When some issues are fixed, you need to create Service B' for the new application version. If you want to keep Service B for external access, you can switch 20% of traffic to Service B'. If the new version stably runs for a period of time, you can switch all traffic from Service B to Service B'. Then, you can delete Service B.



Ingress controllers of ACK provide the following traffic splitting methods to support the preceding requirements of application releases.

- Traffic splitting based on request headers. This method applies to scenarios where canary releases or A/B testing is required.
- Traffic splitting based on cookie. This method applies to scenarios where canary releases or A/B testing is required.
- Traffic splitting based on query parameters. This method applies to scenarios where canary releases or A/B testing is required.
- Traffic splitting based on Service weights. This method applies to scenarios where blue-green releases are required.

Annotations

Ingress controllers use the following annotations to implement canary releases of an application.

- nginx.ingress.kubernetes.io/service-match
  This annotation is used to configure match rules for requests to the new application version.

```
nginx.ingress.kubernetes.io/service-match: |
  <service-name>: <match-rule>
# Parameters
# service-name: the name of a Service. Requests that match the rules specified by match-rule are forwarded to the Service.
# match-rule: the match rules for requests.
#
# Match rules:
# 1. Supported match types
# - header: based on the request header. Regular expressions and exact match rules are supported.
# - cookie: based on the cookie. Regular expressions and exact match rules are supported.
# - query: based on the query parameters. Regular expressions and exact match rules are supported.
#
# 2. Match methods
# - Regular expressions: /{regular expression}/. A regular expression is enclosed within forward slashes (/).
# - Exact match rules:"{exact expression}". An exact match rule is enclosed within quotation marks (").
```

Examples of match rules:

```
# If the value of the foo parameter in the request header matches the regular expression ^bar$, the request is forwarded to the new-nginx Service.
new-nginx: header("foo", /^bar$/)
# If the value of the foo parameter in the request header exactly matches the value bar, the request is forwarded to the new-nginx Service.
new-nginx: header("foo", "bar")
# If the value of the foo parameter in the cookie matches the regular expression ^sticky-.+$, the request is forwarded to the new-nginx Service.
new-nginx: cookie("foo", /^sticky-.+$/)
# If the value of the foo parameter in the query parameters exactly matches the value bar, the request is forwarded to the new-nginx Service.
new-nginx: query("foo", "bar")
```

- nginx.ingress.kubernetes.io/service-weight
  This annotation is used to set the weights of the Services for the earlier and new application versions.

```
nginx.ingress.kubernetes.io/service-weight: |
  <new-svc-name>:<new-svc-weight>, <old-svc-name>:<old-svc-weight>
Parameters:
new-svc-name: the name of the Service for the new application version.
new-svc-weight: the traffic weight of the Service for the new application version.
old-svc-name: the name of the Service for the earlier application version.
old-svc-weight: the traffic weight of the Service for the earlier application version.
```

Example of Service weight configurations:

```
nginx.ingress.kubernetes.io/service-weight: |
  new-nginx: 20, old-nginx: 60
```

Step 1: Create an application

Create an NGINX application and deploy an Ingress controller to enable Layer 7 access to the application by using domain names.

1. Create a Deployment and a Service.

i. Create a file named *nginx.yaml*.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: old-nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   run: old-nginx
 template:
  metadata:
   labels:
    run: old-nginx
  spec:
   containers:
   - image: registry.cn-hangzhou.aliyuncs.com/xianlu/old-nginx
    imagePullPolicy: Always
    name: old-nginx
    ports:
    - containerPort: 80
     protocol: TCP
   restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
 name: old-nginx
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: old-nginx
 sessionAffinity: None
 type: NodePort
```

ii. Run the following command to create the Deployment and Service:

```
kubectl apply -f nginx.yaml
```

2. Create an Ingress.

i. Create a file named *ingress.yaml*.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
spec:
 rules:
 - host: www.example.com
   http:
    paths:
    # Configure a Service for the earlier application version.
    - path: /
      backend:
        serviceName: old-nginx
        servicePort: 80
```

ii. Run the following command to create the Ingress:

```
kubectl apply -f ingress.yaml
```

3. Test access to the Ingress.

i. Run the following command to query the IP address of the Ingress for external access:

```
kubectl get ingress
```

ii. Run the following command to access the Ingress:

```
curl -H "Host: www.example.com" http://<EXTERNAL_IP>
```

The following output is returned:

```
old
```

Step 2: Implement a canary release of the application

Release a new NGINX application version and configure Ingress rules.

1. Create a Deployment and a Service for the new application version.

i. Create a file named *nginx1.yaml*.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: new-nginx
spec:
 replicas: 1
 selector:
  matchLabels:
   run: new-nginx
 template:
  metadata:
   labels:
    run: new-nginx
  spec:
   containers:
   - image: registry.cn-hangzhou.aliyuncs.com/xianlu/new-nginx
    imagePullPolicy: Always
    name: new-nginx
    ports:
    - containerPort: 80
     protocol: TCP
   restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
 name: new-nginx
spec:
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  run: new-nginx
 sessionAffinity: None
 type: NodePort
```

ii. Run the following command to create the Deployment and Service:

```
kubectl apply -f nginx1.yaml
```

2. Configure Ingress rules for the new application version.

   ACK provides the following types of Ingress rules. Select a type of Ingress rule based on your requirements.

   - Configure Ingress rules to forward requests that match the rules to the new application version. In the following example, only requests whose values of the foo parameters in the request headers match the regular expression `bar` are forwarded to the new application version.

a. Modify the Ingress that is created in based on the following content.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
 annotations:
  # Only requests whose values of the foo parameters in the request headers match the regular expre
ssion bar are forwarded to the new-nginx Service.
  nginx.ingress.kubernetes.io/service-match: |
   new-nginx: header("foo", /^bar$/)
spec:
 rules:
 - host: www.example.com
  http:
   paths:
   # Configure a Service for the earlier application version.
   - path: /
    backend:
     serviceName: old-nginx
     servicePort: 80
   # Configure a Service for the new application version.
   - path: /
    backend:
     serviceName: new-nginx
     servicePort: 80
```

b. Test access to the Ingress.

- Run the following command to access the NGINX application:

  ```
  curl -H "Host: www.example.com" http://<EXTERNAL_IP>
  ```

  The following output is returned:

  ```
  old
  ```

- Run the following command to access the NGINX application by using a request whose value of the foo parameter in the request header matches the regular expression `bar`:

  ```
  curl -H "Host: www.example.com" -H "foo: bar" http://<EXTERNAL_IP>
  ```

  The following output is returned:

  ```
  new
  ```

  You can run the preceding commands again to test the access. The result is that only requests whose values of the foo parameters in the request headers match the regular expression `bar` are forwarded to the new application version.

- Configure Ingress rules to forward a specific proportion of requests that match the rules to the new application version. In the following example, only 50% of the requests whose values of the foo parameters in the request headers match the regular expression `bar` are forwarded to the new version.

a. Modify the Ingress that is created in based on the following content.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
 annotations:
  # Only requests whose values of the foo parameters in the request headers match the regular expre
ssion bar are forwarded to the new-nginx Service.
  nginx.ingress.kubernetes.io/service-match: |
    new-nginx: header("foo", /^bar$/)
  # Only 50% of the requests that match the preceding rule are forwarded to the new-nginx Service.
  nginx.ingress.kubernetes.io/service-weight: |
    new-nginx: 50, old-nginx: 50
spec:
 rules:
 - host: www.example.com
  http:
   paths:
   # Configure a Service for the earlier application version.
   - path: /
    backend:
     serviceName: old-nginx
     servicePort: 80
   # Configure a Service for the new application version.
   - path: /
    backend:
     serviceName: new-nginx
     servicePort: 80
```

b. Test the access to the Ingress.

- Run the following command to access the NGINX application:

  ```
  curl -H "Host: www.example.com" http://<EXTERNAL_IP>
  ```

  The following output is returned:

  ```
  old
  ```

- Run the following command to access the NGINX application by using a request whose value of the foo parameter in the request header matches the regular expression `bar`:

  ```
  curl -H "Host: www.example.com" -H "foo: bar" http://<EXTERNAL_IP>
  ```

  The following output is returned:

  ```
  new
  ```

You can run the preceding commands again to test the access. The result is that only 50% of the requests whose values of the foo parameters in the request headers match the regular expression `bar` are forwarded to the new application version.

- Configure Ingress rules to forward a specific proportion of requests to the new NGINX application. In the following example, only 50% of requests are forwarded to the new NGINX application.

a. Modify the Ingress that is created in based on the following content.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
 annotations:
   # 50% of requests are forwarded to the new-nginx Service.
   nginx.ingress.kubernetes.io/service-weight: |
     new-nginx: 50, old-nginx: 50
spec:
 rules:
 - host: www.example.com
   http:
    paths:
    # Configure a Service for the earlier application version.
    - path: /
      backend:
       serviceName: old-nginx
       servicePort: 80
    # Configure a Service for the new application version.
    - path: /
      backend:
       serviceName: new-nginx
       servicePort: 80
```

b. Run the following command to access the Ingress:

```
curl -H "Host: www.example.com" http://<EXTERNAL_IP>
```

You can run the preceding command again to test the access. The result is that only 50% of the requests are forwarded to the new NGINX application.

Step 3: Delete the earlier application version and the related Service

If the new NGINX application runs as expected for a period of time, you need to bring the earlier application version offline and provide only the new application version for access.

1. Modify the Ingress that is created in based on the following content.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: gray-release
spec:
 rules:
 - host: www.example.com
   http:
    paths:
    # Configure a Service for the new application version.
    - path: /
      backend:
       serviceName: new-nginx
       servicePort: 80
```

2. Run the following command to access the Ingress:

```
curl -H "Host: www.example.com" http://<EXTERNAL_IP>
```

The following output is returned:

```
new
```

You can run the preceding command again to test the access. The result is that all the requests are forwarded to the new NGINX application.

3. Delete the Deployment and Service for the earlier NGINX application.
   i. Run the following command to delete the Deployment for the earlier NGINX application:

   ```
   kubectl delete deploy <Deployment name>
   ```

   ii. Run the following command to delete the Service for the earlier NGINX application:

   ```
   kubectl delete svc <Service name>
   ```

14.2. Manage releases by using Helm

Container Service for Kubernetes (ACK) integrates the package manager Helm to help you deploy applications to the cloud in an efficient manner. You can install a chart multiple times in a Kubernetes cluster. Each time you install a chart, a release is created. Therefore, the management of release versions is required. To meet this requirement, ACK provides the release feature that allows you to manage applications that are deployed by using Helm in the ACK console.

Prerequisites

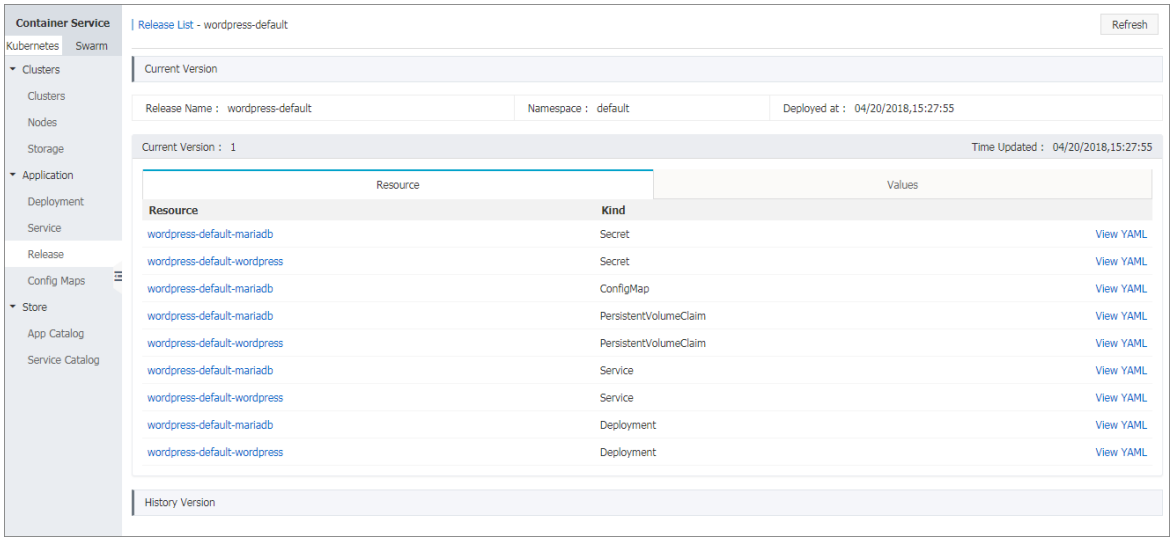
- A Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群.
- An application is deployed by using Helm from App Catalog or Service Catalog. For more information, see Simplify Kubernetes application deployment by using Helm. In this example, an application named tf-model is deployed.

View release details

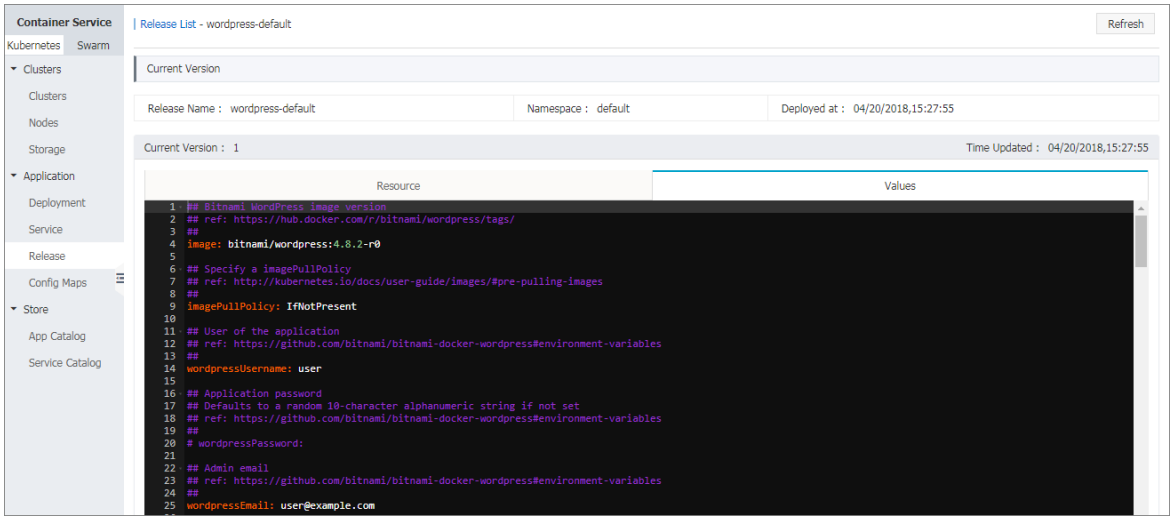
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Helm**.
5. You can view the details of a release. In this example, find tf-model and click **View Details** in the Actions column. The details page of tf-model appears.

   You can view information about tf-model, such as the current version and history versions. In this example, the current version is 1 and none history version exists. You can also view information about the resources of tf-model, such as the names, types, and YAML files.

   > **Note** To view the status of a resource, click the name of the resource to go to the Kubernetes dashboard page.

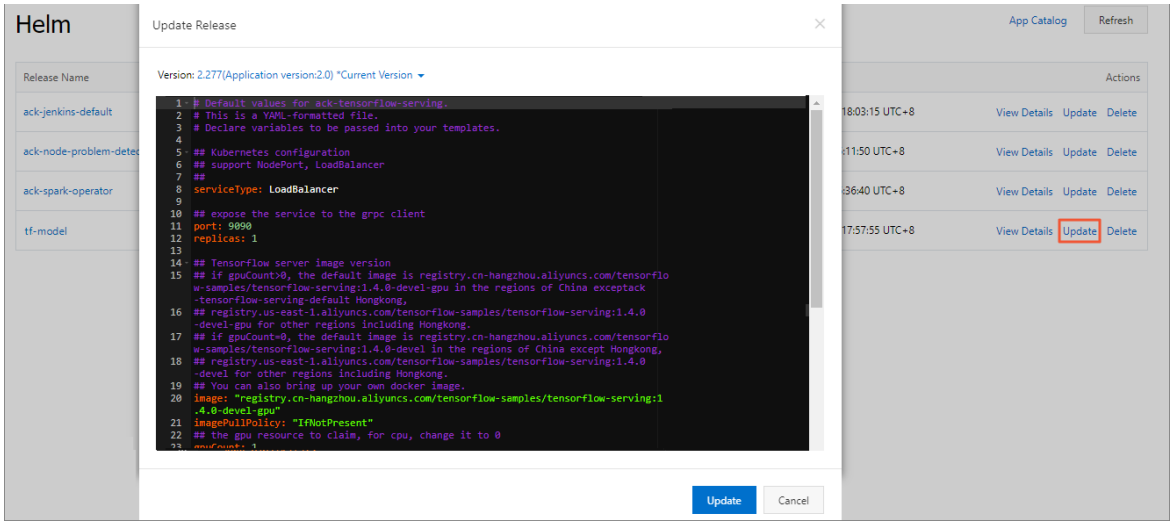


6. Click the **Parameters** tab. You can view the parameters of the Helm chart.

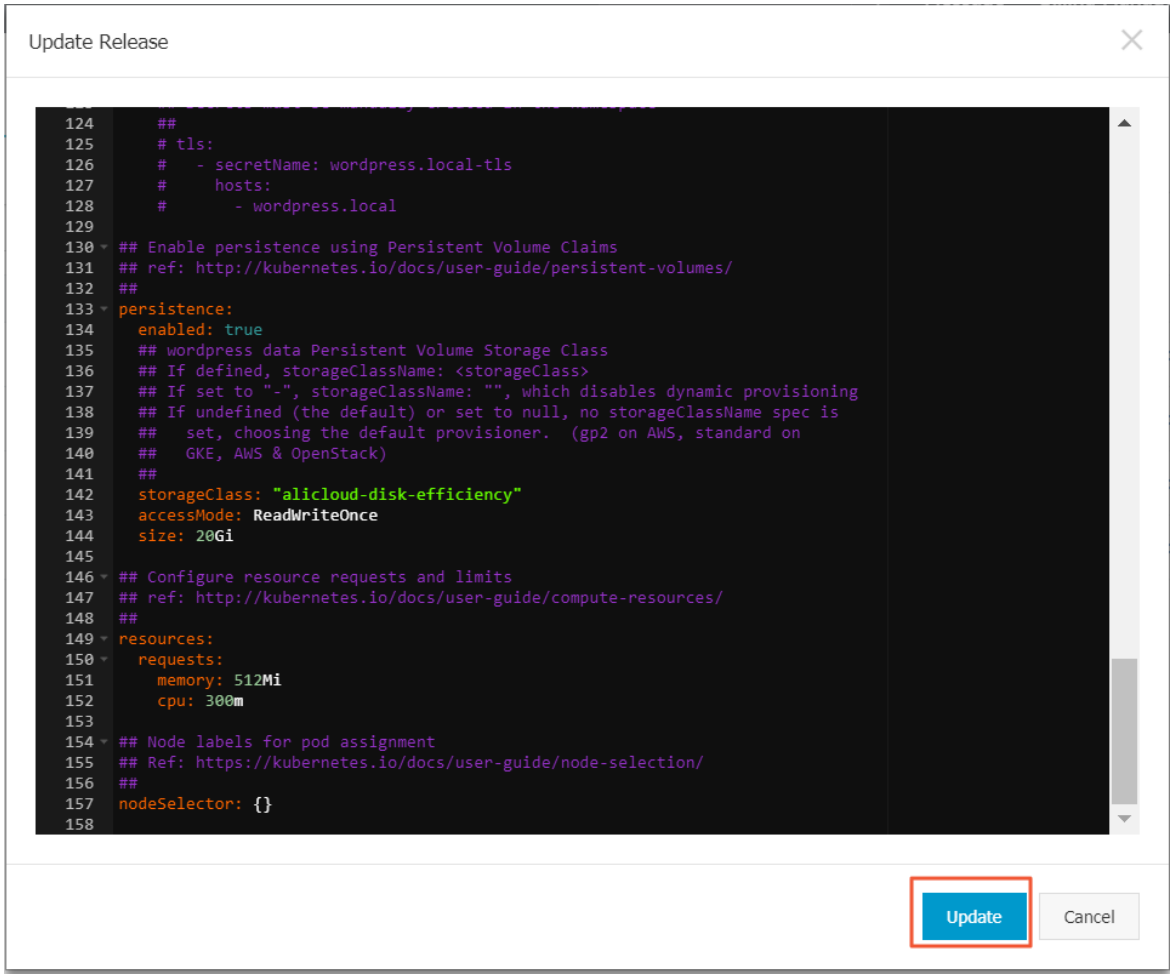


Update a release

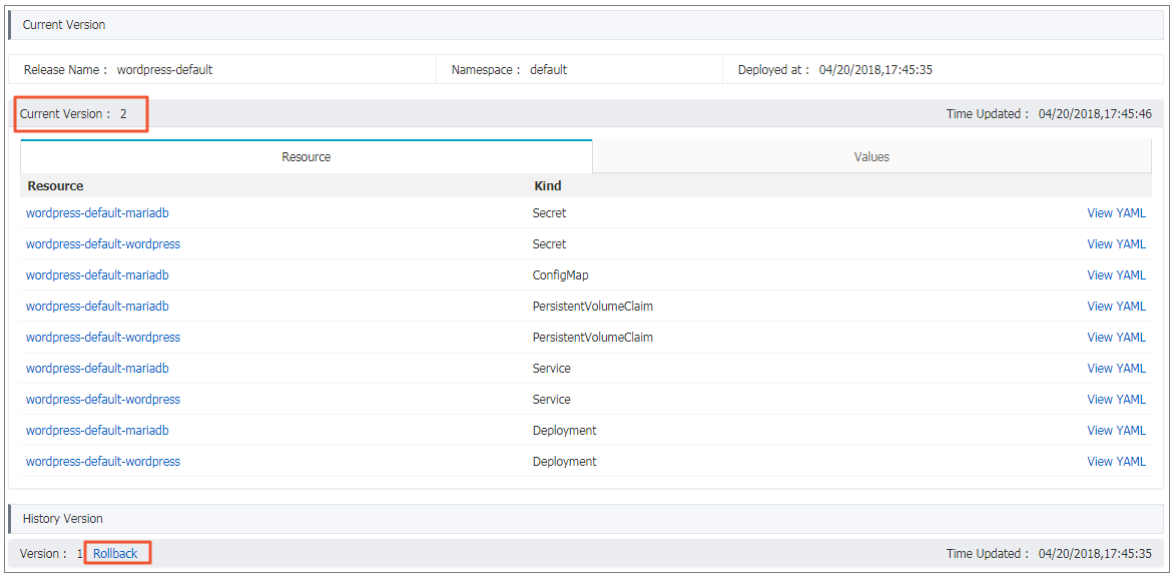
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, click **Releases**.
5. Click the **Helm** tab. The list of releases appears.
6. You can update a release. In this example, find tf-model and click **Update** in the Actions column. The Update Release dialog box appears.



7. In the **Update Release** dialog box, modify the parameters based on your requirements and click **Update**.

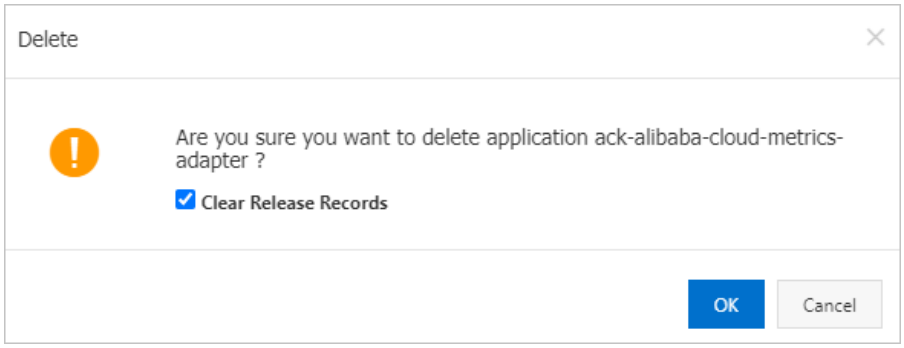


On the details page of tf-model, the current version of tf-model changes to 2. In the History Version section, you can find the history version 1. To roll back to version 1, click **Roll Back**.



Delete a release

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, click **Releases**.
5. Click the **Helm** tab. The list of releases appears.
6. You can delete a release. In this example, find tf-model release and click **Delete** in the Actions column.
7. In the **Delete** dialog box, select **Clear Release Records** and click **OK**. After you the release is deleted, the resources of the release are deleted, including the Services and Deployments.



Note If you clear **Clear Release Records**, the deletion of tf-model does not take effect. You can still find tf-model in the release list. If you create another release named tf-model, a release name conflict occurs.

15.Knative

15.1. Overview

Knative is a Kubernetes-based serverless framework. Knative creates a cloud-native and cross-platform orchestration standard for serverless applications. Knative enforces this standard by streamlining the creation of container or function, workload management and auto scaling, and event models.

Knative architecture

The following figure shows how different roles interact with each other in the Knative architecture.

- Developers
  Developers of serverless applications can call the Kubernetes-native API to deploy serverless applications based on Knative.
- Contributors
  Contributors in the preceding figure mainly refer to the contributors to the Knative community.
- Operators
  Knative can be integrated with environments that are supported by Knative, such as environments of cloud service providers or internal environments of enterprises. Knative is based on Kubernetes. Therefore, it can be deployed in environments where Kubernetes is implemented.
- Users
  Users access services through Istio gateways, or run serverless applications by triggering Knative events.

Core Knative components

Knative consists of three core components. These components form a general-purpose serverless framework:

- Tekton: provides the capability to create images from source code.
  Tekton is used to retrieve source code from the code repository, compile the code into images, and push the images to the image repository. All these operations are performed in Kubernetes pods.
- Knative Eventing: provides event management capabilities, such as producing and consuming events.

Knative Eventing has designed an all-in-one eventing system for event-driven serverless applications. The eventing system provides features that allow you to install external event sources, register and subscribe to events, and filter events. The event system decouples event producers and event consumers. An event producer can generate events before active event consumers listen to events. An event consumer can listen to events before active producers produce events.

- Knative Serving: manages the workloads of serverless applications. Knative Serving enables automatic scaling for pods where serverless applications are deployed based on Knative events and user requests. If no workload is processed, the number of pods is scaled to zero.
  Knative Serving is used to manage the workloads of serverless applications that provide services to users. The most important feature of Knative Serving is automatic scaling. The scaling capacity is not limited. Knative Serving also supports canary release.

Knative add-ons

Knative supports third-party add-ons. The GitHub add-on component is supported. This component provides support for GitHub event sources.

Deploy Knative components in a Kubernetes cluster

Knative components can be deployed in standard managed Kubernetes clusters, standard dedicated Kubernetes clusters, and serverless Kubernetes (ASK) clusters. You can deploy Knative components by using the following methods:

- To deploy Knative components in ACK standard managed clusters and standard dedicated ACK clusters, see Deploy Knative.
- You can deploy Knative components in serverless Kubernetes clusters. For more information, see Enable Knative.

> **Note** Make sure that the version of the cluster is 1.16 or later.

Knative billing rules

Knative is free of charge. However, you are charged for the cloud resources that are created during the use of Knative. For example, you may create ECS instances, SLB instances, and NAT gateways. You are charged for creating and using these resources. For more information about the billing methods of these resources, see the following topics:

- ECS billing overview
- Billing overview

Related information

- Deploy Knative
- Use Knative to deploy serverless applications

15.2. Knative version release notes

15.2.1. Knative 0.18.3

Container Service for Kubernetes (ACK) supports Knative 0.18.3. This topic describes the changes and features of Knative 0.18.3.

> **Note** We recommend that you set the apiVersion parameter of Knative Services to V1. The V1alpha1 and V1beta1 versions will be deprecated after Knative 0.19.0 is released.

Features

- To use the features of Knative 0.18.3, the Kubernetes version must be 1.18 or later.
- Supports multiple containers in a pod. Knative Services allow you to deploy multiple containers in a pod.
- Supports header-based routing policies. You can specify the `Knative-Serving-Tag: {revision-tag}` header in a request. This allows the Kourier ingress to directly send requests to specific revisions. You can use this feature to implement header-based canary releases.
- Adds compatibility with the nodeSelector feature in Kubernetes. The following parameters are required to configure this feature: affinity, nodeSelector, and tolerations.
- Knative Services support the dry-run feature. This feature allows you to validate the configurations of the current revision template. You can use one of the following parameters to enable the dry-run feature in the template:
  - features.knative.dev/podspec-dryrun: enabled
  - features.knative.dev/podspec-dryrun: strict

> **Note** When you create a Knative Service:
>
> - If you set features.knative.dev/podspec-dryrun to `enabled`, a dry run is performed if Kubernetes supports the dry-run feature. If Kubernetes does not support the dry-run feature, the system still tries to create the Knative Service.
> - If you set features.knative.dev/podspec-dryrun to `strict`, a `failure` is returned if Kubernetes does not support the dry-run feature.

15.3. Manage Knative components

15.3.1. Deploy Knative

Knative is a Kubernetes-based serverless framework. The main objective of Knative is to develop a cloud-native and cross-platform orchestration standard for serverless applications. This topic describes how to deploy Knative in a Container Service for Kubernetes (ACK) cluster.

Prerequisites

创建Kubernetes托管版集群.

> **Notice**
>
> - Knative 0.14.0 and later support only clusters of Kubernetes 1.15 and later. The following types of ACK clusters are supported: standard managed Kubernetes clusters, standard dedicated Kubernetes clusters, and serverless Kubernetes (ASK) clusters.
> - If a standard managed Kubernetes cluster or a standard dedicated Kubernetes cluster is used, the cluster must contain at least three worker nodes.

Procedure

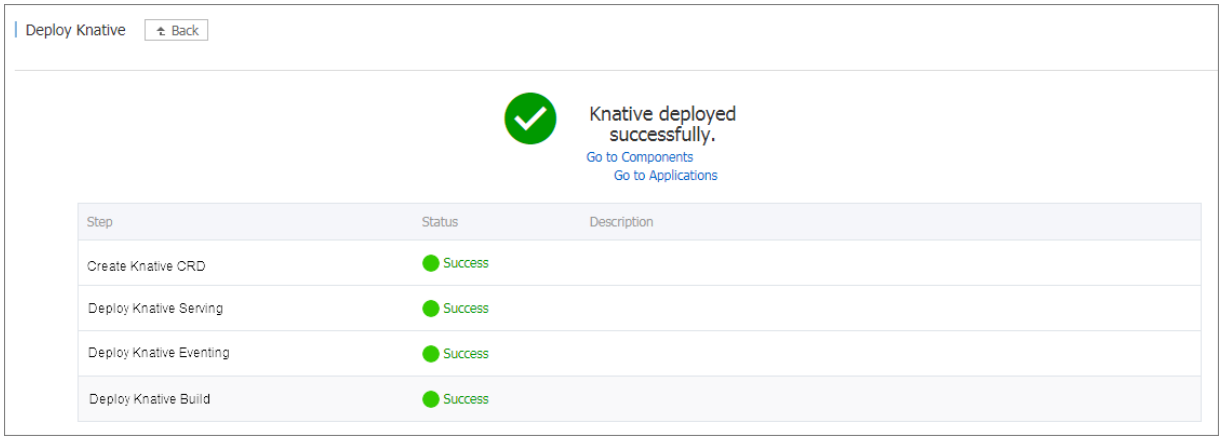
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Applications > Knative**.

5. On the **Components** tab, click **Deploy Knative**.
6. Select the Knative components that you want to install and click **Deploy**.
   - **Serving**: manages serverless applications. Knative Serving enables automatic scaling for pods where serverless applications are deployed based on Knative events and user requests. If no workload is processed, the number of pods is scaled to zero.
   - **Eventing**: provides event management capabilities, such as producing and consuming events.
   - **Tekton**: provides a flexible approach to create images from source code.

Result

After Knative is deployed, you can perform the following steps to check the deployment result:

- Click **Go to Components** to view information about the components.
- Click **Go to Applications** to view information about operations related to Knative applications.



15.3.2. Deploy a Knative component

If you have skipped a component when you deploy Knative, you can go to the **Components** tab to install the skipped component. Container Service for Kubernetes (ACK) allows you to deploy core Knative components and add-ons. This topic describes how to deploy a Knative component in a Container Service for Kubernetes (ACK) cluster.

Prerequisites

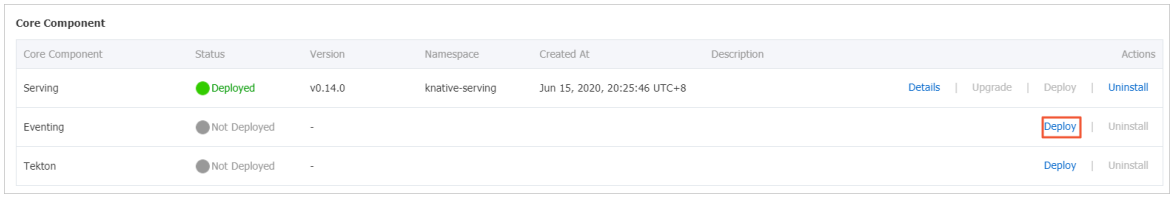
- 创建Kubernetes托管版集群
- Deploy Knative

Context

If you have skipped components when you deploy Knative, perform the following steps to deploy these components:

Procedure

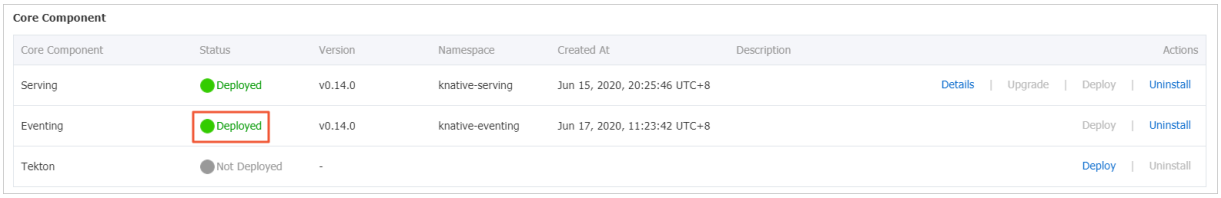
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Knative**.
5. Find the Knative components that display **Not Deployed** in the **Status** column and click **Deploy** in the Actions column.



6. In the message that appears, click **Confirm**.

Result

After the components are deployed, go to the **Components** tab and verify that these components are in the *Deployed* state.



Related information

- Deploy Knative
- Upgrade a Knative component

15.3.3. Upgrade a Knative component

Container Service for Kubernetes (ACK) allows you to upgrade Knative components. You can use Knative Serving to manage serverless applications. This enables auto scaling of pods based on Knative events and user requests. If no workload is processed, the number of pods is scaled to zero. This topic describes how to upgrade Knative Serving.

Prerequisites

- Knative Serving is deployed. For more information, see Deploy a Knative component.
- The Kubernetes version is later than 1.15.0.
- Only Knative Serving 0.11.0 can be upgraded.

> **Note** We recommend that you upgrade Knative Serving during off-peak hours.

Context

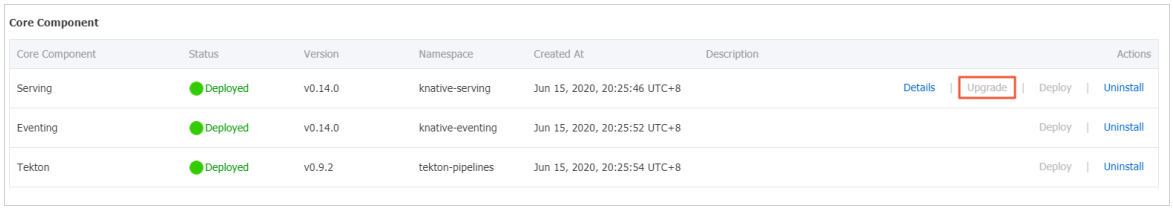
You can upgrade Knative Serving to V0.14.0.
Knative Serving 0.14.0 provides the following features:

- By default, a minimum of 20 latest Knative Service revisions are retained. The default retention period is 48 hours.
- Highly reliable subcomponents are provided, such as controller and hpaautoscaler.

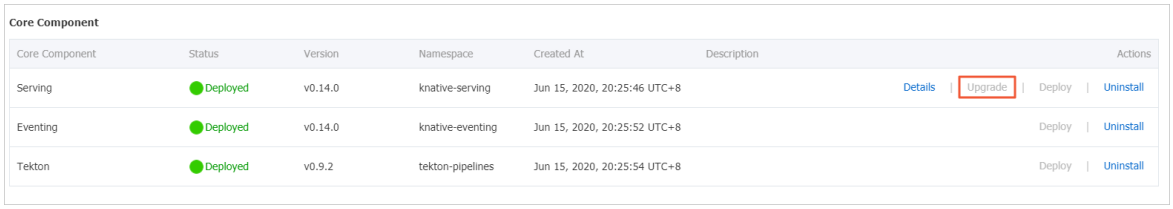
Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Knative**.

5. On the **Components** tab, find **Serving** in the **Core Component** section and click **Upgrade**.



After Knative Serving is upgraded, the result is displayed, as shown in the following figure.



15.3.4. Collect log data of Knative components

You can collect the log of Knative components and perform operations analysis and troubleshooting based on the collected log. This topic describes how to collect the Docker stdout log of Knative components by using Log Service.

Prerequisites

创建Kubernetes托管版集群

Context

Knative components include:

- knative-serving
  - activator
  - autoscaler
  - autoscaler-hpa
  - controller
  - webhook
- knative-eventing
  - eventing-controller
  - eventing-webhook

Procedure

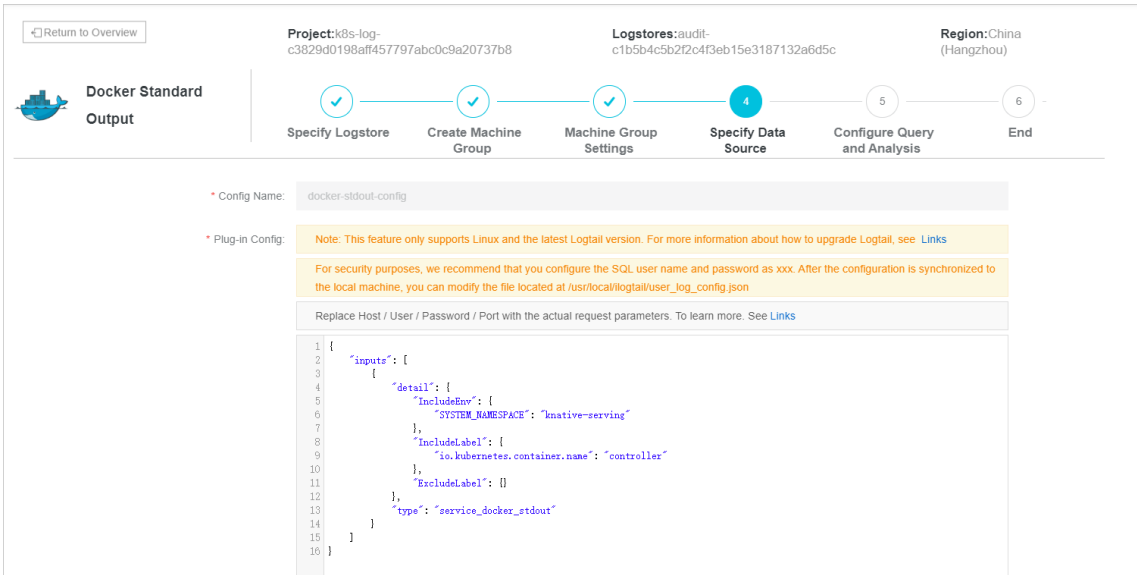
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, click **Cluster Information** and click the **Cluster Resources** tab.
5. On the **Cluster Resources** tab, click the hyperlink next to **Log Service Project**.
6. In the upper-right corner of the **Overview** page of the Log Service project, click **Import Data**.
7. In the **Import Data** dialog box, click **Docker Standard Output - Container**.
8. Complete the Docker standard output settings.

   The following content describes how to complete the **Configure Data Source** wizard page. For more

information, see Configure log collection.

The following sample code provides an example on how to configure the **data source** to collect the stdout log of controller in the knative-serving namespace. Sample code:

```
{
  "inputs": [
    {
      "detail": {
        "IncludeEnv": {
     "SYSTEM_NAMESPACE":"knative-serving"
    },
        "IncludeLabel": {
     "io.kubernetes.container.name": "controller"
        },
        "ExcludeLabel": {}
      },
      "type": "service_docker_stdout"
    }
  ]
}
```



Note

- In the IncludeEnv section, set SYSTEM_NAMESPACE to the namespace where controller is installed.
- In the IncludeLabel section, set io.kubernetes.container.name to the name of the component, which is controller in this example.

9. In the **Log Query** section, click **Try Now** to view the collected log.

15.3.5. Configure alerts for Knative components

You can use Prometheus Monitoring in Application Real-Time Monitoring Service (ARMS) to collect metrics of Knative components. This topic describes how to configure alerts for Knative components.

Context

The following metrics of Knative components are collected:

- Number of ready pods where the components are deployed.
- CPU usage of the pods where the components are deployed.
- Memory usage of the pods where the components are deployed.

Knative components include:

- knative-serving
  - activator
  - autoscaler
  - autoscaler-hpa
  - controller
  - webhook
- knative-eventing
  - eventing-controller
  - eventing-webhook

Step 1: Install the Prometheus Monitoring component

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab. Then, find and click **ack-arms-prometheus**.
4. On the **App Catalog - ack-arms-prometheus** page, select the cluster for which you want to enable ARMS Prometheus in the **Deploy** section, and click **Create**.

> **Note** By default, **Namespace** and **Release Name** are set to **arms-prom**.

Step 2: View pod monitoring information

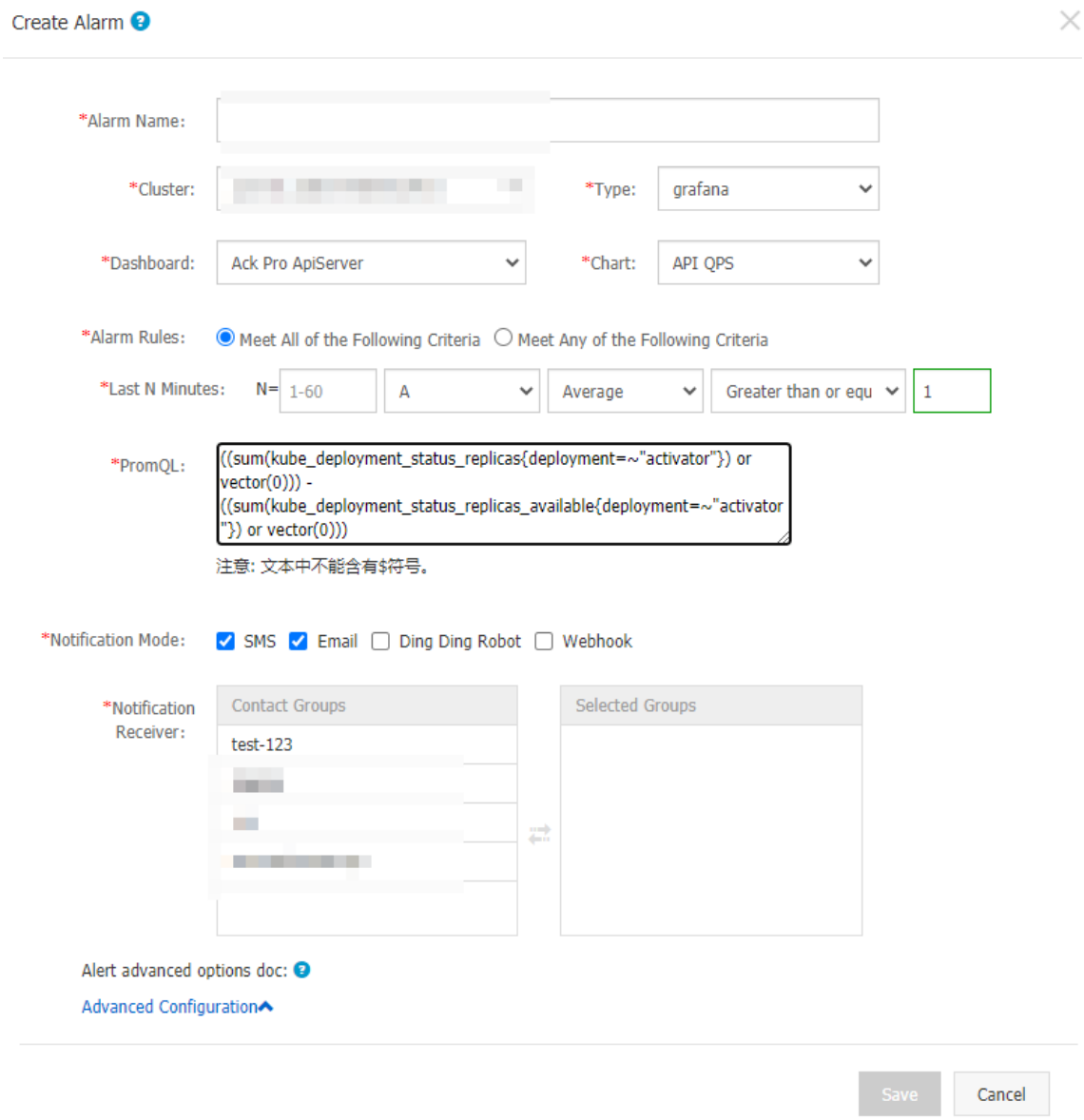
Before you set an alert policy for a pod where the components are deployed, check the monitoring information about the pod.

1. Log on to the ARMS console.
2. In the left-side navigation pane, click **Prometheus Monitoring**.
3. On the **Prometheus Monitoring** page, click **k8s state** in the **Installed Dashboards** column. On the page that appears, you can view the number of pods, the CPU usage of the pods, and the memory usage of the pods.

Step 3: Set alert policies for Knative components

1. You can select one of the two available methods to go to the Create Alarm dialog box.
   - On the **New DashBoard** page of the Prometheus Grafana dashboard, click the icon to go to the ARMS Prometheus **Create Alarm** dialog box.
   - In the left-side navigation pane of the console, choose **Alerts > Alert Policies**. On the **Alarm Policies** page, choose **Create Alarm > Prometheus** in the upper-right corner.
2. In the **Create Alarm** dialog box, set the parameters.

   The activator component of Knative Serving is used in this example. If the number of ready pods is less than the specified number, an alert is triggered. In this case, the number of unavailable pods is greater than or equal to one. The following figure shows the alert policy.



PromQL queries for the activator component:

```
((sum(kube_deployment_status_replicas{deployment=~"activator"}) or vector(0))) - ((sum(kube_deployment_status_replicas_available{deployment=~"activator"}) or vector(0)))
```

For more information, see Create an alert.

3. Click **Save**.

15.3.6. Uninstall Knative components

On the **Components** tab, you can uninstall the Knative components that you have deployed. Container Service for Kubernetes (ACK) allows you to uninstall core Knative components and add-ons. This topic describes how to uninstall Knative components. **Knative Eventing** is used as an example.

Prerequisites

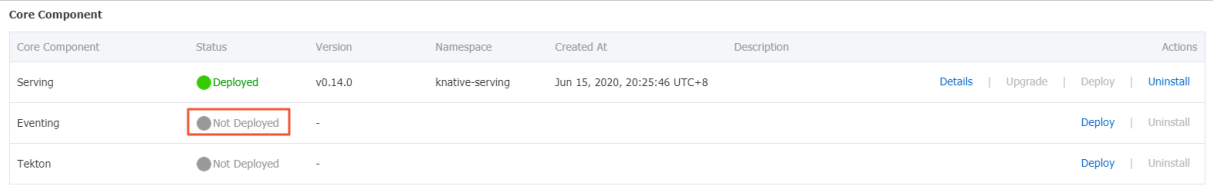
- 创建Kubernetes托管版集群
- Deploy Knative

Procedure

1. Log on to the ACK console.
2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. In the left-side navigation pane of the details page, choose **Applications > Knative**.
4. On the **Components** tab, select the component that you want to uninstall and click **Uninstall**.
5. In the message that appears, click **Confirm**.

Result

After the component is uninstalled, go to the **Components** tab and verify that the state of **Eventing** is *Not Deployed*.



15.3.7. Uninstall Knative from an ACK cluster

This topic describes how to uninstall Knative from a Container Service for Kubernetes (ACK) cluster.

Prerequisites

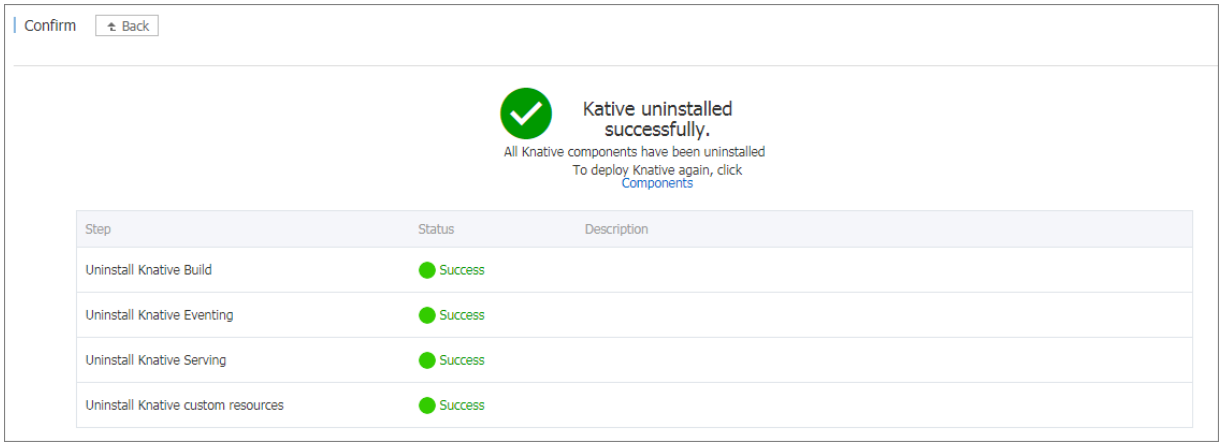
Deploy Knative

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Knative**.
5. On the **Components** tab, click **Uninstall** in the upper-right corner.
6. In the **Confirm** message, select **I confirm that I have read the above information and want to uninstall Knative** and click **OK**.

Result

After Knative is uninstalled, the result is displayed, as shown in the following figure.



15.4. Manage Knative services

15.4.1. Use Knative to deploy serverless applications

Knative Services are used to deploy applications. This topic describes how to create a Knative Service.

Prerequisites

- 创建Kubernetes托管版集群
- Deploy Knative
- Knative Serving is deployed. For more information, see Deploy a Knative component.

Step 1: Deploy a Knative Service

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Knative**.
5. On the **Services** tab, click **Create Service** in the upper-right corner.
6. Configure the Knative Service.

Parameter	Description
Namespace	Select the namespace to which the Service belongs.
Service Name	Enter a name for the Service.
Image Name	To select an image, click Select Image. In the dialog box that appears, select an image and click **OK**. You can also enter the address of an image in a private registry. The image address must be in the following format: *domainname/namespace/imagename:tag*. *registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go* is used in this example.

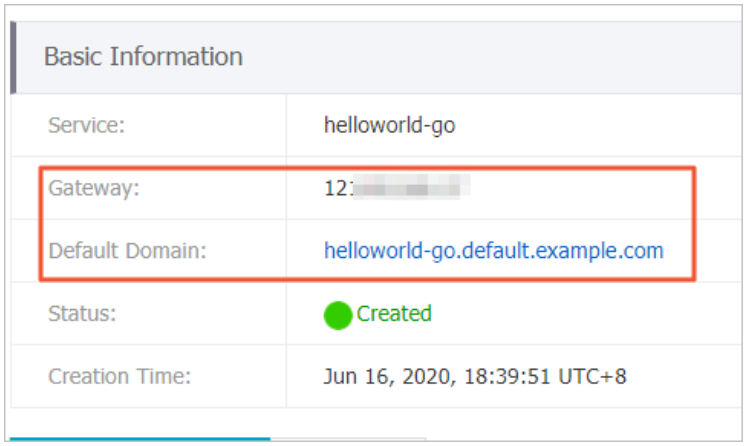
Parameter	Description
Image Version	To select an image version, click Select Image Version. *73fbdd56* is selected in this example. 
Access Protocol	Both **HTTP** and **gRPC** are supported.
Container Port	The container port that you want to expose. The port number must be in the range of 1 to 65535.
Internal Access Only	If you select **Internal Access Only**, the Knative Service cannot be accessed over the Internet.
Maximum Concurrent Requests	Set the maximum number of concurrent requests supported by the container. The default value is 0. This indicates that the number of concurrent requests is unlimited.
Minimum Pods	Set the minimum number of pods that must be kept running when no request is received. If you set this parameter to 0, the number of running pods is reduced to zero when no request is received.
Maximum Pods	Set the maximum number of pods that are allowed to run.
Resource Limit	Set the maximum amount of CPU, memory, and GPU resources that can be allocated to the Knative Service. This prevents the Service from occupying an excess amount of resources. CPU usage is measured in cores. Memory usage is measured in bytes or megabytes.
Lifecycle	Includes the Command and Parameter parameters. Take note of the following limits: ○ If Command and Parameter are not set, the default values of Command and Parameter of the image are used. ○ If you set only Parameter, the default value of Command of the image and the specified value of Parameter are used. ○ If you set both Command and Parameter, the default values of Command and Parameter of the image are overwritten by the specified values.
Environment Variables	Set environment variables in key-value pairs.
Volume	You can mount local storage volumes and persistent volume claims (PVCs) to the container. ○ Add Local Storage: You can select HostPath, ConfigMap, Secret, or EmptyDir. The specified volume is mounted to a path in the container. For more information, see Volumes. ○ Add PVC: You can select Cloud Storage.

7. Click **Create**.

Step 2: Access the Knative Service

After the Knative Service is created, point the domain name of the Knative Service to the IP address of the access gateway by modifying the hosts file. Then, you can use the domain name to access the Knative Service. Perform the following steps:

1. On the **Services** tab, click the name of the Service.
   In the **Basic Information** section, you can view information about the access gateway and domain name.

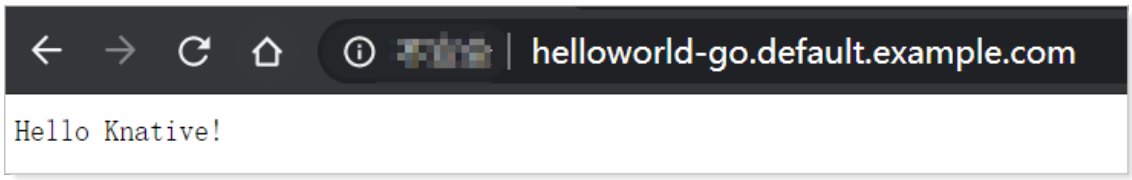


2. Modify the hosts file to point the domain name of the Knative Service to the IP address of the gateway.

   Example:

```
121.xx.xxx.xx helloworld-go.default.example.com
```

3. After you modify the hosts file, use the domain name to access the Knative Service.



15.4.2. Create a revision

You can upgrade a Knative Service by adding a revision to Knative. This topic describes how to create a revision for a Knative Service. If you want to roll back from a newly released application version to an earlier version, you only need to switch inbound traffic to the application of the specified version.

Prerequisites

- 创建Kubernetes托管版集群
- Deploy Knative
- Deploy a Knative component
- A Knative Service is deployed.

Context

When you deploy a Knative Service, the system creates a revision named stock-service-example-v1 and forwards all inbound traffic to stock-service-example-v1.

Procedure

1. Log on to the ACK console.

2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Knative**.
5. On the **Services** tab, select the namespace, find the Service, and then click **Details** in the Actions column.
6. In the upper-corner of the details page, click **Create Revision**.
7. On the **Create Revision** page, set the required parameters.
   i. Configure the parameters on the **Basic Information** wizard page.
   For more information about how to configure the basic settings of a revision, see Set revision parameters.
   ii. Configure the parameters on the **Traffic Splitting Settings** wizard page.
   - Revisions: A revision is created each time a Knative Service is released.
   - Percent %: The percentage of traffic that is forwarded to a revision. The total sum of the traffic percentage values specified for all revisions must be 100%.
8. Click **Create**.
   On the details page of the Knative Service, you can view information about the newly created revision on the **Revision Information** tab.

Additional information

After a revision is created, you can delete the revision.

1. Log on to the ACK console.
2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. In the left-side navigation pane of the details page, choose **Applications > Knative**.
4. On the **Services** tab, select the namespace to which the Service belongs and click the name of the Service.
5. In the **Revision Information** section, find the revision that you want to delete and click **Delete** in the **Actions** column.
6. In the **Delele Revision** message, click **OK**.

15.4.3. Set a custom domain name for Knative Serving

This topic describes how to set a custom domain name for Knative Serving.

Prerequisites

- A domain name is registered. For more information, see Alibaba Cloud Domains.
- Knative is deployed in your Container Service for Kubernetes (ACK) cluster. For more information, see Deploy Knative.

Context

Knative Serving uses the default domain name `example.com` in a route. The default format of a fully qualified domain name for a Knative Serving route is `{route}.{namespace}.{default-domain}`.

Use kubectl to modify a domain name

The following example demonstrates how to change the default domain name to a custom one in the *knative-serving* namespace.

1. Connect to Kubernetes clusters by using kubectl.
2. Run the following command to modify the ConfigMap named *config-domain*.

   ```
   kubectl edit cm config-domain --namespace knative-serving
   ```

3. Modify the ConfigMap.

   Replace example.com with the custom domain name that you want to use, and save the configuration. In this example, a custom domain name mydomain.com is used.

   ```
   apiVersion: v1
   data:
    mydomain.com: ""
   kind: ConfigMap
   [...]
   ```

Deploy a Knative Service

If you have deployed a Knative Service, Knative automatically updates all Knative Services and routes based on the ConfigMap.

1. Deploy a Knative Service. In this example, a Knative Service named *helloworld-go* is deployed. For more information, see Use Knative to deploy serverless applications.
2. View the deployment result.
   - For Knative 0.7, run the following command to query the domain name of the Knative Service:

     ```
     kubectl get route ${SVC_NAME} --output jsonpath="{.status.url}"| awk -F/ '{print $3}'`
     ```

   - For Knative 0.6, run the following command to query the domain name of the Knative Service:

     ```
     kubectl get route helloworld-go --output jsonpath="{.status.domain}"
     ```

     If the following output is returned, it indicates that the custom domain name is applied:

     ```
     helloworld-go.default.mydomain.com
     ```

Map the custom domain name to the Knative Service

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Knative**.
5. On the **Services** tab, click the name of the Service.
   In the **Basic Information** section, you can view information about the access gateway and domain name.
6. Add the IP address of the gateway to the resolution settings of the custom domain name in the Alibaba Cloud DNS console.

Result

Run the following command to query the result:

```
curl http://helloworld-go.default.mydomain.com
```

15.4.4. Delete a revision

This topic describes how to delete a revision.

Prerequisites

- 创建Kubernetes托管版集群
- Deploy Knative
- Deploy Knative Serving.
- Use Knative to deploy serverless applications

Procedure

1. Log on to the ACK console.
2. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
3. In the left-side navigation pane of the details page, choose **Applications > Knative**.
4. On the **Services** tab, select the namespace to which the Service belongs and the click the Service.
5. In the **Revision Information** section, find the revision that you want to delete and click **Delete** in the **Actions** column.
6. In the **Delete Revision** message, click **OK**.

15.4.5. Delete a Knative Service

This topic describes how to delete a Knative Service.

Prerequisites

- 创建Kubernetes托管版集群
- Deploy Knative
- Knative Serving is deployed

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Knative**.
5. On the **Services** tab, find the Service that you want to delete and click **Delete** in the **Actions** column.
6. In the **Delete Service** message, click **OK**.

15.4.6. Deploy a canary release for a Knative Service

Knative supports canary releases. You can deploy a canary release to split traffic between two application versions based on a specified ratio. This topic describes how to deploy a canary release for a Knative Service.

Prerequisites

Deploy Knative:

- If you want to deploy Knative in a Container Service for Kubernetes (ACK) cluster, see Deploy Knative.
- If you want to deploy Knative in a serverless Kubernetes (ASK) cluster, see Enable Knative.

Step 1: Create a Knative Service

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Knative**.
5. On the **Services** tab, click **Create Service** in the upper-right corner.
6. On the Create Service page, set the **Namespace** and **Service Name** parameters. Then, select an image and an image version.

Parameter	Description
Namespace	Select the namespace to which the Service belongs.
Service Name	Enter a name for the Service. In this example, *helloworld-go* is used.
Image Name	To select an image, click **Select Image**. In the dialog box that appears, select an image and click **OK**. You can also enter the address of a private image **registry**. The registry address must be in the *domainname/namespace/imagename:tag* format. In this example, *registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go* is used.
Image Version	Click **Select Image Version** and select an image version. By default, the **latest** version is used. In this example, *73fbdd56* is selected.
Access Protocol	**HTTP** and **gRPC** are supported. **Note** gRPC is developed based on the HTTP/2 standard and Protocol Buffers (protobuf) serialization protocol, and supports various programming languages. Compared with HTTP/1.1, HTTP/2 allows you to send and receive packets more efficiently.
Container Port	The container port that you want to expose. The port number must be from 1 to 65535.

   For more information about the other parameters, see Create a Knative Service.
7. Click **Create**.

   After the Service is created, you can find the Service on the **Services** tab.
8. Run the following command to access the Service:

   ```
   curl -H "<Default domain>" http://<Gateway address>
   ```

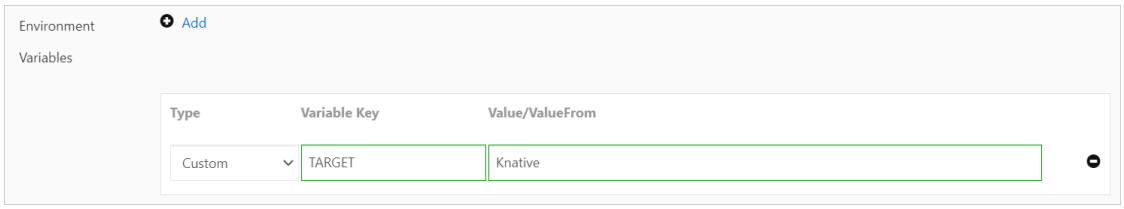
   - Default domain: In this example, "host: helloworld-go.default.example.com" is used.
   - Gateway address: In this example, "39.106.XX.XX" is used.

   Expected output:

   ```
   Hello World!
   ```

Step 2: Create a revision to deploy a canary release

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Applications > Knative**.
5. Create a revision.
   i. On the **Services** tab, select the Service that you created and click **Details** in the **Actions** column.
   ii. Click **Create Revision**.
   iii. On the **Basic Information** wizard page, click **Advanced** and add the following environment variable setting: `TARGET=Knative`.



   iv. Click **Next Step**.
   v. On the **Traffic Splitting Settings** wizard page, set the **Percent %** parameter of the revision to 0. Then, click **Create**.

> **Note** The sum of traffic ratios of all revisions must be 100.

   vi. After the revision is created, you can find details about the new version on the **Services** tab.
   vii. Run the following command to access the Service:

> **Note** The **Percent** parameter is set to 0 for the new version. This means that all requests to access the helloworld-go Service are sent to the earlier version.

```
curl -H "host: helloworld-go.default.example.com" http://39.106.XX.XX
```

   Expected output:

```
Hello World!
```

6. Modify the traffic splitting ratio to deploy a canary release.
   i. On the **Services** tab, select the Service that you created and click **Details** in the **Actions** column.
   ii. Click **Split Traffic**.
   iii. In the **Split Traffic** dialog box, set the **Percent** parameter to 50% for both the earlier and new versions. Then, click **OK**.
   iv. After you change the values of the **Percent** parameters, check the details about the old and new versions on the **Services** tab.

v. Run the following command to access the Service:

```
curl -H "host: helloworld-go.default.example.com" http://39.106.XX.XX
```

Expected output:

The **Percent** parameter is set to 50% for both the earlier and new versions. This means that requests to access the helloworld-go Service are evenly distributed to the earlier and new version. You can change the value of the **Percent** parameter to implement canary releases. The new version is completely released if you set the **Percent** parameter to 100% for the new version. During the process, you can change the value of the **Percent** parameter to roll back if issues are found in the new version.

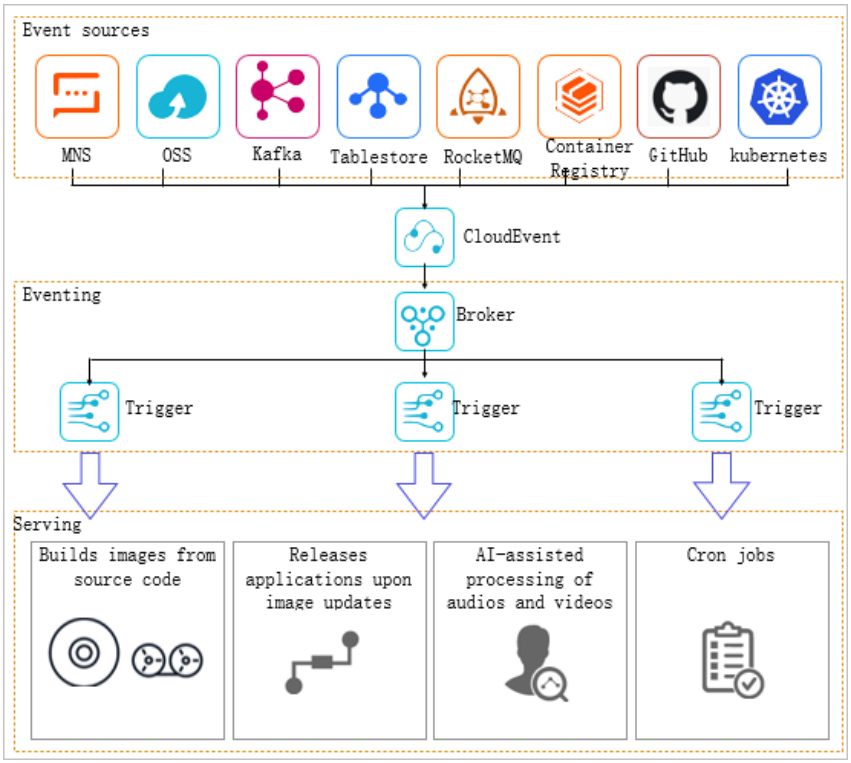
15.5. Knative event processing

15.5.1. Overview

Knative Eventing is designed to address common demands for cloud-native development. It allows you to bind event sources with event consumers to handle events. This topic describes event sources, event handling, and event consumption of Knative Eventing.

Features

Knative Eventing meets the common needs in cloud-native development. In addition, Knative Eventing provides an architecture for serverless event-driven mode. The architecture contains event sources, event ingesting and subscription, and event filtering. The following figure shows the event-driven architecture.



- Event sources
  - Open source Knative provides various event sources such as Kafka and GitHub.
  - Open source Knative also allows you to use cloud services as event sources. The cloud services include Message Service (MNS) and RocketMQ.
- Event handling
  - Knative Eventing routes events from brokers to event sinks or consumers. You can create one or more triggers to filter or subscribe to specific events.
  - Events can be consumed by serverless applications that are managed by Knative.
- Event consumption
  - Automatically releases applications when images in Container Registry are updated.
  - Automatically creates images upon code submission.
  - Supports cron jobs and AI-assisted processing of audios and videos.

How to start

For more information about how to use Knative Eventing, see Deploy a Knative component.

Related information

- Use Knative to manage GitHub events
- Use Knative to manage MnsOss event sources

15.5.2. Use Knative to manage GitHub events

This topic describes how to use Knative to manage GitHub events.

Prerequisites

- Knative Serving, Knative Eventing, and the GitHub add-on are deployed in the Container Service for Kubernetes (ACK) cluster. For more information, see Deploy a Knative component.
- A custom domain name is configured in Knative. For more information, see Set a custom domain name for Knative Serving.

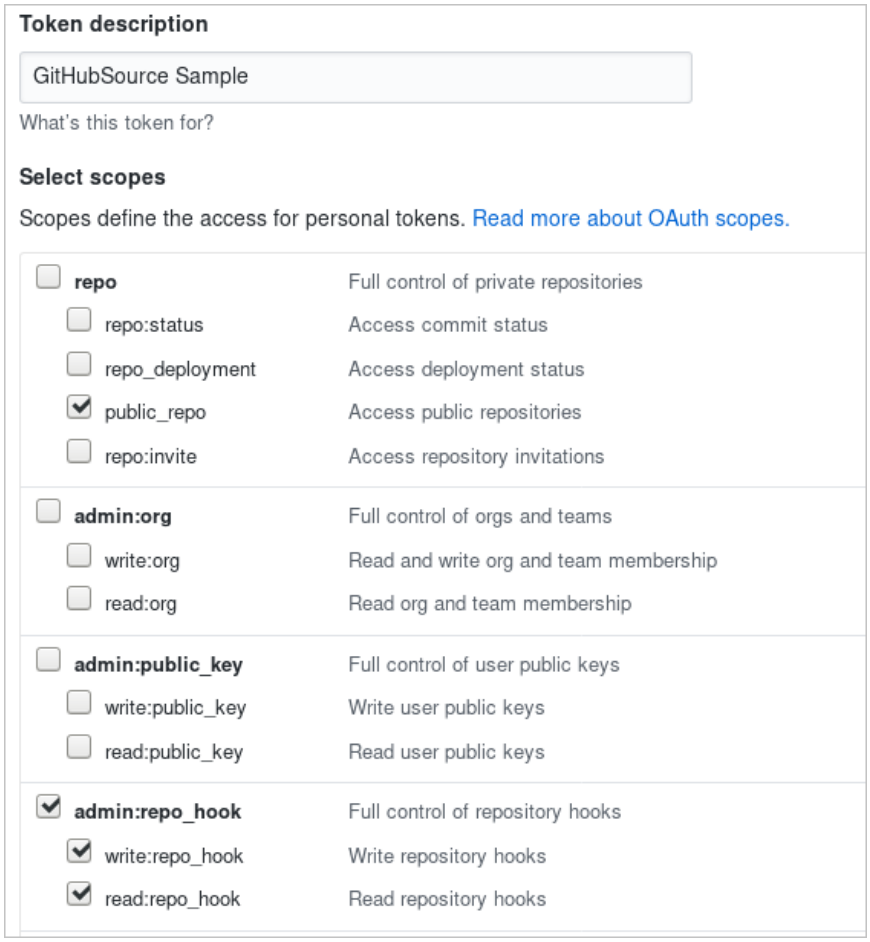
Step 1: Create a Knative Service

For more information, see Use Knative to deploy serverless applications.

Step 2: Create a GitHub token

1. Create a personal access token.

   A personal access token is required when you call the GitHub API. For more information, see Personal access tokens.
   The following figure shows how to create a token named *GitHubSource Sample*.



   > **Note** To trigger events from your public repositories and create webhooks for these repositories, select `repo:public_repo` and `admin:repo_hook`.
   > You can enter a custom token name.

2. Connect to Kubernetes clusters by using kubectl.
3. Use the following method to generate a random string as secretToken:

```
head -c 8 /dev/urandom | base64
```

4. Update the *githubsecret.yaml* file.

   If the generated token is *personal_access_token_value*, you must set the secretToken field. The following code block is an example:

```
apiVersion: v1
kind: Secret
metadata:
 name: githubsecret
type: Opaque
stringData:
 accessToken: personal_access_token_value
 secretToken: asdfasfdsaf
```

5. Run the following command to create a GitHub token:

```
kubectl --namespace default apply githubsecret.yaml
```

Step 3: Create a GitHub event source

You can create a GitHub event source to receive events generated by GitHub.

1. Connect to Kubernetes clusters by using kubectl.
2. Create a file named *github-source.yaml* and copy the following content to the file:

```
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: GitHubSource
metadata:
 name: githubsourcesample
spec:
 eventTypes:
  - pull_request
 ownerAndRepository: <YOUR USER>/<YOUR REPO>
 accessToken:
  secretKeyRef:
   name: githubsecret
   key: accessToken
 secretToken:
  secretKeyRef:
   name: githubsecret
   key: secretToken
 sink:
  apiVersion: serving.knative.dev/v1alpha1
  kind: Service
  name: github-event-display
```

3. Run the following command to create a GitHub event source in the default namespace:

```
kubectl --namespace default apply github-source.yaml
```

Result

In the GitHub repository, choose **Settings > Webhooks** and check whether a verified webhook URL is displayed.

> **Note** The domain name must have an Internet Content Provider (ICP) number. Otherwise, the domain name is inaccessible.

Run the following command in the GitHub repository to create a `pull request`. Then, an event is triggered.

```
kubectl --namespace default get pods
kubectl --namespace default logs github-event-display-XXXX user-container
```

In Knative Eventing, you can view event details that are similar to the following content:

```
2018/11/08 18:25:34 Message Dumper received a message: POST / HTTP/1.1
Host: github-event-display.knative-demo.svc.cluster.local
Accept-Encoding: gzip
Ce-Cloudeventsversion: 0.1
Ce-Eventid: a8d4cf20-e383-11e8-8069-46e3c8ad****
Ce-Eventtime: 2018-11-08T18:25:32.819548012Z
Ce-Eventtype: dev.knative.source.github.pull_request
Ce-Source: https://github.com/someuser/somerepo/pull/1
Content-Length: 21060
Content-Type: application/json
User-Agent: Go-http-client/1.1
X-B3-Parentspanid: b2e514c3dbe94c03
X-B3-Sampled: 1
X-B3-Spanid: c85e346d89c8be4e
X-B3-Traceid: abf6292d458fb8e7
X-Envoy-Expected-Rq-Timeout-Ms: 60000
X-Envoy-Internal: true
X-Forwarded-For: 12*.*.*.*, 12*.*.*.*
X-Forwarded-Proto: http
X-Request-Id: 8a2201af-5075-9447-b593-ec3a243a****
{"action":"opened","number":1,"pull_request": ...}
```

15.5.3. Use Knative to manage MnsOss event sources

You can collect Object Storage Service (OSS) events from MnsOss event sources and manage the events accordingly. This is suitable for scenarios where facial recognition is required. This topic describes how to use Knative to manage MnsOss event sources.

Prerequisites

- Knative Serving and Knative Eventing are installed. For more information, see Deploy Knative.
- An OSS bucket is created in the OSS console. For more information, see Create buckets.
- Message Service (MNS) is activated. For more information, see Activate MNS and authorize RAM users to access MNS.

Step 1: Deploy MnsOss

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the details page of the cluster, choose **Applications > Knative**.
5. On the **Components** tab, find MnsOss and click **Deploy** in the **Actions** column.
6. In the **Deploy MnsOss** message, click **Confirm**.

Step 2: Configure OSS event notification settings

1. Log on to the OSS console.
2. Click **Buckets**, and then click the name of the target bucket.
3. In the Event Notification section, click **Configure**. On the page that appears, click **Create Rule**.
4. In the left-side navigation pane, choose **Basic Settings > Event Notification**.
5. In the **Create Rule** panel, configure the rule parameters. The parameters are described in the following table:

Parameter	Description
Rule Name	Specify the name of the event notification rule. The name of an event notification rule can contain only letters, digits, and hyphens (-). The name cannot exceed 85 characters in length.
Events	Select one or more events that can trigger the event notification rule from the drop-down list. For example, if you select CopyObject, the event notification rule is triggered when specific objects are created or overwritten by copying objects. If multiple event notification rules apply to the same object, the values of this parameter in these rules must be different. For example, if you configure Events as **CopyObject** when you create an event notification rule for objects whose names contain the *examplefolder* prefix, you cannot select **CopyObject** as the value for Events when you create another event notification rule for objects whose names contain the same prefix. For more information about the events, see Events.
Resource Description	Specify the objects to which the event notification rule applies. ○ **Full Name**: Enter the complete path of the object to which the event notification rule applies. Example: *examplefolder/myphoto.jpg*. ○ **Prefix and Suffix**: Enter the prefix and suffix of the objects to which the event notification rule applies. The following examples show how to configure the prefix and suffix: ■ To create a rule that applies to all objects in the bucket, leave Prefix and Suffix empty. ■ To create a rule that applies to all objects in the *examplefolder* folder in the root folder of the bucket, set Prefix to **examplefolder/** and leave Suffix empty. ■ To create a rule that applies to all JPG objects in the bucket, leave Prefix empty and set Suffix to **.jpg**. ■ To create a rule that applies to all MP3 objects in the *examplefolder* folder in the root folder of the bucket, set Prefix to **examplefolder/** and Suffix to **.mp3**. To create a Resource Description, click **Add**. You can create up to five **Resource Description** entries.

Parameter	Description
Endpoint	Configure the endpoint to which notifications are sent. Valid values: **HTTP** and **Queue**. ◦ **HTTP**: Enter the address of the HTTP endpoint to which notifications are sent. Example: `http://198.51.100.1:8080`. For more information about how to enable an HTTP endpoint, see Manage topics and HttpEndpoint. ◦ **Queue**: Enter the name of an MNS queue. For more information about how to create a queue, see Create a queue. To create an endpoint, click **Add**. You can create up to five endpoints.

6. Click **OK**.

   After you configure the OSS event notification settings, a topic is created on the **Topics** page in the MNS console.

Step 3: Create an MNS token

1. Log on to the MNS console.
2. In the left-side navigation pane, click **Topics**.
3. In the top navigation bar of the MNS console, select the region where your cluster is deployed.
4. On the **Topics** page, click the topic that you want to manage.
5. On the **Topics** page, click **Get Endpoint** in the upper-right corner of the page.
6. In the **Endpoint** section of the **Topic Details** page, copy the public endpoint.
7. Obtain an AccessKey ID and AccessKey secret of the current account. For more information, see How can I obtain an AccessKey ID and AccessKey secret?.
8. Run the following command to encode the public endpoint, AccessKey ID, and AccessKey secret by using Base64. Then, a token is generated.

   ```
   echo '{ "url":"https://xxxx.mns.cn-shanghai.aliyuncs.com/", "accessKeyId":"xxx","accessKeySecret":"xx" }' | base64
   ```

9. Create a Secret to store and manage the token.
   i. Create a file named *mnsoss-secret.yaml*.

   ```
   apiVersion: v1
   kind: Secret
   metadata:
    name: mnsoss-secret
   type: Opaque
   data:
    mns: eyAidXJsIjoiaHR0cHM6Ly94eHh4Lm1ucy5jbi1zaGFuZ2hhaS5hbGl5dW5jcy5jb20vIiwgImFjY2Vzc0tleUlkIjoieHh4IiwiYWNjZXNzS2V5U2VjcmV0IjoieHgiIH0K
   ```

   Replace the value of `mns` with the token that is generated in Step .

   ii. Run the following command to create a Secret:

   ```
   kubectl apply -f mnsoss-secret.yaml
   ```

Step 4: Create a service account and a service broker

1. Create a service account.

i. Create a file named *mnsoss-sa.yaml*.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: eventing-sources-mnsoss
subjects:
- kind: ServiceAccount
 name: mnsoss-sa
 namespace: default
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: eventing-sources-mnsoss-controller
---
apiVersion: v1
kind: ServiceAccount
metadata:
 name: mnsoss-sa
```

ii. Run the following command to create a service account:

```
kubectl apply -f mnsoss-sa.yaml
```

2. Run the following command to create a service broker:

```
kubectl label namespace default knative-eventing-injection=enabled
```

Step 5: Create an MnsOss event source

To receive MnsOss events, you must create an MnsOss event source.

1. Create a file named *mnsoss-source.yaml*.

```
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: MnsOssSource
metadata:
 labels:
  controller-tools.k8s.io: "1.0"
 name: mnsoss-face
spec:
 # Add fields here
 serviceAccountName: mnsoss-sa
 accessToken:
  secretKeyRef:
   name: mnsoss-secret
   key: mns
 sink:
  apiVersion: eventing.knative.dev/v1alpha1
  kind: Broker
  name: default
 topic: mns-en-topics-oss-face-image-2381221888dds9129
```

Set `topic` to the name of the topic that is generated in the MNS console.

2. Run the following command to create an MnsOss event source:

```
kubectl apply -f mnsoss-source.yaml
```

Step 6: Create a Knative Service

Create a Knative Service to verify whether the created MnsOss event source works as normal. A Knative Service named event-display is created in this example.

1. Create a file named *service.yaml*.

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
 name: event-dispaly
 namespace: default
spec:
 template:
  spec:
   containers:
   - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/event-display:1215
```

2. Run the following command to create a Knative Service:

```
kubectl apply -f service.yaml
```

Step 7: Create a trigger

Create a trigger to subscribe to OSS events.

1. Create a file named *trigger.yaml*.

```
apiVersion: eventing.knative.dev/v1alpha1
kind: Trigger
metadata:
 name: oss-trigger
 namespace: default
spec:
 subscriber:
  ref:
   apiVersion: serving.knative.dev/v1alpha1
   kind: Service
   name: event-dispaly
```

2. Run the following command to create a trigger:

```
kubectl apply -f trigger.yaml
```

Check results

When you upload files to the OSS bucket, notifications are sent to pods.

```
2020/12/16 13:04:19 receive cloudevents.Event:
{"events": [{
        "eventName": "ObjectCreated:PostObject",
        "eventSource": "acs:oss",
        "eventTime": "2019-06-18T06:44:16.000Z",
        "eventVersion": "1.0",
        "oss": {
          "bucket": {
            "arn": "acs:oss:cn-beijing:1041208914252405:testjian",
            "name": "testjian",
            "ownerIdentity": "1041208914252405",
            "virtualBucket": ""},
          "object": {
            "deltaSize": 0,
            "eTag": "137138904F2E18D307D04EB38EA44CDA",
            "key": "timg.jpg",
            "size": 12990},
          "ossSchemaVersion": "1.0",
          "ruleId": "demo-i****"},
        "region": "cn-beijing",
        "requestParameters": {"sourceIPAddress": "42.120.7*.***"},
        "responseElements": {"requestId": "5D08884070BC12B192C6****"},
        "userIdentity": {"principalId": "104120891425****"}}] "137138904F2E18D307D04EB38EA44CDA",
            "key": "timg.jpg",
            "size": 12990},
          "ossSchemaVersion": "1.0",
          "ruleId": "demo-i****"},
        "region": "cn-beijing",
        "requestParameters": {"sourceIPAddress": "42.120.7*.***"},
        "responseElements": {"requestId": "5D08884070BC12B192C6****"},
        "userIdentity": {"principalId": "104120891425****"}}]}
```

15.6. Best practices

15.6.1. Configure alerting on Knative

This topic describes how to use Log Service to collect log data from Knative and configure alerts based on the collected log data.

Prerequisites

- A Knative Service is deployed. For more information, see Use Knative to deploy serverless applications.
-

Procedure

1. Configure search conditions.

   For more information, see 分析概述.

   i. Log on to the Log Service console and click the name of the Log Service project that you want to manage. On the **Logstores** tab, find the Logstore created in Step 1.

   ii. Click the name of the created Logstore.

iii. Enter the following command into the search box below the Logstore name and click **Search & Analyze** on the right side of the page.

To configure an alert based on the number of errors that have occurred, enter the following SQL statement:


```
* | select 'ERROR' , count(1) as total group by 'ERROR'
```

2. Configure alert settings.

For more information, see Create an alert rule.

i. Click **Save as Alert** in the upper-right corner of the page.

> **Note** In the Trigger Condition field, enter a conditional expression that determines whether to trigger an alert. For more information, see Syntax of trigger conditions in alert rules. In this example, set **Search Period** to 1 minute, **Check Frequency** to 1 minute, and **Trigger Condition** to *total > 3*. This means that a search is performed every other minute. If three or more errors are found from the log data within 1 minute, an alert is triggered.

ii. In the **Alert Monitoring Rule** pane, specify the configurations of the alert rule.

Parameter	Description
Rule Name	The name of the alert rule. The name must be 1 to 64 characters in length.
Check Frequency	The time interval at which the server checks log data. **Note** The server collects and checks only the first 100 data entries that are generated during the time specified by the search period.
Dashboard	Select or create a dashboard.
Chart Name	The name of the chart. The name must be 1 to 64 characters in length.
Query	Enter an SQL statement.
Search Period	The **Search Period** parameter specifies the time range of log data that the server reads when a search is performed. You can select a relative time or a time frame. For example, if you set **Search Period** to 15 minutes (relative) and start the query at 14:30:06, Log Service reads the log data that was written from 14:15:06 to 14:30:06. If you set **Search Period** to 15 minutes (time frame) and start the query at 14:30:06, Log Service reads the log data that was written from 14:15:00 to 14:30:00.

Parameter	Description
Trigger Condition	The conditional expression that determines whether to trigger an alert. If the condition is met, Log Service sends notifications based on the settings of **Check Frequency** and **Notification Interval**. For example, you can enter `pv%100 > 0 && uv > 0` into the Trigger Condition field. **Note** In the conditional expression, you can use `$[serial number]` to differentiate charts. For example, you can use `$0` to identify the chart whose serial number is 0. For more information, see How can I view the serial number of a chart?
Trigger Threshold	When the cumulative number of times that the trigger condition is met reaches the threshold, notifications are sent based on the **notification interval**. The count does not increase if the trigger condition is not met. The default value of **Trigger Threshold** is 1. This means that each time the **trigger condition** is met, Log Service checks the **notification interval** to decide whether to send a notification. You can also set Log Service to send only one notification after the trigger condition is met a specified number of times. For example, if you set the value to 100, Log Service checks the **notification interval** to decide whether to send a notification only when the trigger condition is met 100 times. If both the **trigger threshold** and **notification interval** are reached, Log Service sends a notification. After Log Service sends the notification, it resets the count to 0. If Log Service fails to check log data due to exceptions such as network failures, the overall count does not change.
Notification Interval	The time interval at which Log Service sends notifications. If the cumulative number of times that the trigger condition is met reaches the **trigger threshold** and the length of time since the last notification reaches the specified notification interval, Log Service sends a notification to the specified contacts. If you set this parameter to 5 minutes, you receive at most one notification every 5 minutes. The default value is None. **Note** You can use the Trigger Threshold and Notification Interval parameters to control the number of notifications that you can receive during a specified time period.
Notification Policy	Select one or more notification methods. You can choose text messages, phone calls, email, DingTalk notifications, webhooks, or notifications from Alibaba Cloud Message Center.

iii. After you configure the notification policy, click **OK**.

Notification method	Description
DingTalk	Sends notifications through DingTalk. When an alert is triggered, a DingTalk chatbot sends a notification to a DingTalk group. To use this notification method, you must specify **Request URL** and **Content**. Enter the content of the DingTalk notification into the **Content** field. The content must be 1 to 500 characters in length. Template variables are supported. For more information about how to configure the DingTalk chatbot and obtain the **request URL**, see Create an alert rule. **Note** Each DingTalk chatbot can send up to 20 notifications per minute.
WebHook	Sends notifications to a custom webhook URL in a specific method. To use this notification method, you must specify **Request URL**, **Request Method**, and **Content**. Valid values of **Request Method**: GET, PUT, POST, DELETE, and OPTIONS. Enter the notification content into the **Content** field. The content must be 1 to 500 characters in length. Template variables are supported.
Alibaba Cloud Message Center	Sends notifications to specific contacts by using the notification method specified in Alibaba Cloud Message Center. To use this notification method, you must specify the **notification content**. Enter the notification content into the **Content** field. The content must be 1 to 500 characters in length. Template variables are supported. In addition, you must specify contacts and the notification method in **Alibaba Cloud Message Center**.

3. Run the following command to access the Hello World application.

   An alert is triggered:

   ```
   curl -H "Host: helloworld-go.default.example.com" http://112.124.XX.XX
   ```

   ```
   Hello Knative!
   ```

4. You are notified by email if setting up email notifications.

Conclusion

After you configure alerts in Log Service, you can detect anomalies of applications and then send notifications to operations and maintenance (O&M) engineers and developers in real time. This ensures the continuity of your business.

15.6.2. Collect application logs

Log Service is an all-in-one service to manage log data. Log Service allows you to collect, consume, ship, query, and analyze log data without development work. After Log Service is enabled on Knative, you can manage and maintain serverless applications with higher efficiency.

Prerequisites

- Log Service is activated. For more information, see Collect log files from containers by using Log Service.
- A Knative Service is deployed. For more information, see Use Knative to deploy serverless applications.

Procedure

1. Collect log data of the Helloworld application.

   For more information, see Use the console to collect Kubernetes stdout and stderr logs in DaemonSet mode.

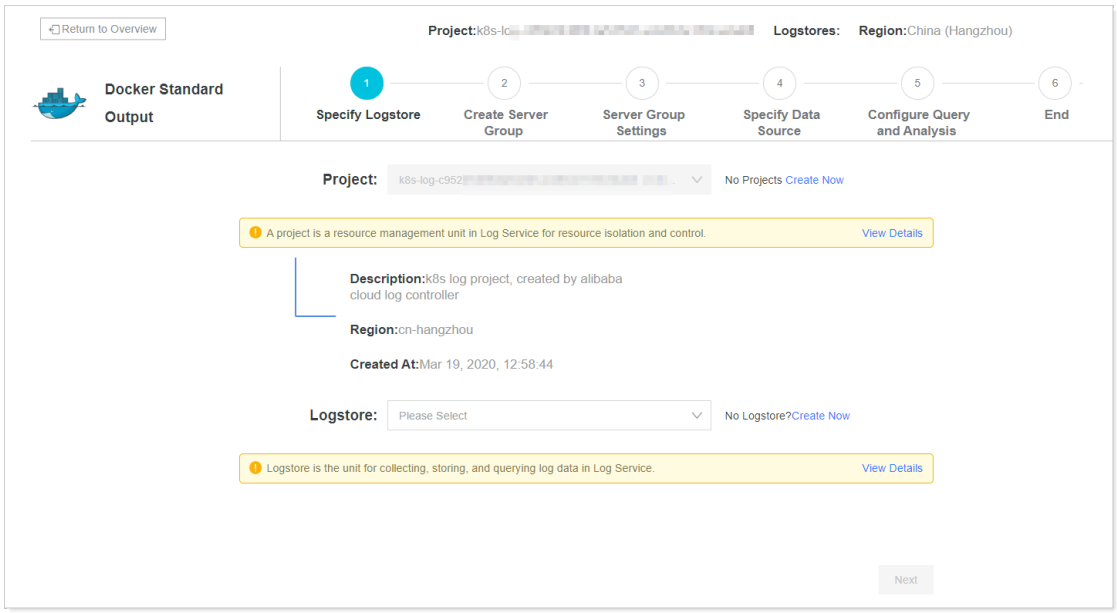
   i. Log on to the Log Service console.

   ii. In the **Projects** section, click **Create Project** to create a Log Service project named helloworld. For more information, see Manage a Logstore.

   iii. Click the created helloworld project.

   iv. Navigate to the details page of the project. In the upper-right corner of the page, click **Import Data**.

   v. In the **Import Data** dialog box, find and click **Docker Standard Output** to navigate to the **Docker Standard Output** page.



   vi. In the **Specify Logstore** step, select the created Logstore and click **Next**.

   vii. Configure **Create Machine Group**.

   If the Logtail agent is installed, click **Use Existing Machine Groups**.
   You can also click *Kubernetes Cluster (Recommended)* to install the Logtail agent or click *Standard Docker Container* to deploy a Logtail container. For more information, see Collect log files from containers by using Log Service and Collect logs from standard Docker containers.

   viii. Click **Complete Installation**.

   ix. Configure **Machine Group Settings**.

x. Configure **Specify Data Source**.

In the plug-in configuration, specify the following environment variable for log collection: `"K_SERVICE": "helloworld-go"`. Use processors to process log data, for example, `"Keys": ["time","level", "msg"]`. The following code block is an example:

```
{
 "inputs": [
  {
   "detail": {
    "IncludeEnv": {
     "K_SERVICE": "helloworld-go"
    },
    "IncludeLabel": {},
    "ExcludeLabel": {}
   },
   "type": "service_docker_stdout"
  }
 ],
 "processors": [
  {
   "detail": {
    "KeepSource": false,
    "NoMatchError": true,
    "Keys": [
     "time",
     "level",
     "msg"
    ],
    "NoKeyError": true,
    "Regex": "(\\d+-\\d+-\\d+\\s+\\d+:\\d+:\\d+)\\s+(\\w+)\\s+(.*)",
    "SourceKey": "content"
   },
   "type": "processor_regex"
  }
 ]
}
```

xi. After the data source is configured, click **Next**.

xii. In the **Configure Query and Analysis** step, enable **Full Text Index** and set key/value index attributes.

xiii. Click **Next** to proceed to the **End** page.

2. Run the following command to access the Helloworld application.

The following log data is generated:

```
curl -H "Host: helloworld-go.default.example.com" http://112.124.XX.XX
```

Expected output:

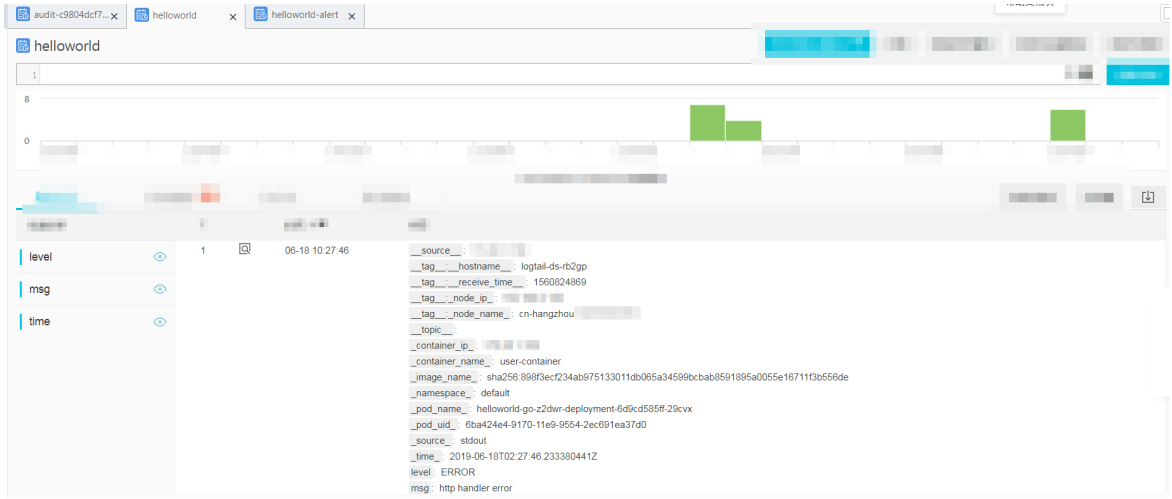
```
Hello Knative!
```

3. Go to the details page of the helloworld project. On the details page, find the created **Logstore**, move the pointer over the icon on the right side of the Logstore, and choose > **Search & Analysis**.

On the page that appears, you can view the log data in the Logstore.

4. Configure search conditions.

   For more information, see 分析概述.
   You can modify **Column Settings** to specify the columns that are displayed. In the following example, three columns are displayed: level, msg, and time.



Conclusion

The preceding example demonstrates how to use Log Service to collect the log data of a serverless Knative application. You can use Log Service to collect and analyze log data of Knative applications. This helps manage and maintain serverless Knative applications in a production environment.

15.6.3. Use elastic container instances in Knative

You can create elastic container instances for a Knative Service. Then, you can use Knative and Virtual Kubelet to create pods on elastic container instances based on your requirements. This topic describes how to create an elastic container instance for a Knative Service by using the following methods.

Prerequisites

- Deploy Knative
- Step 1: Deploy ack-virtual-node in ACK clusters
- Connect to Kubernetes clusters by using kubectl

Method 1: Directly create an elastic container instance for a Knative Service

- If the virtual-node-affinity-injection=enabled label is not added to the namespace, you can create an elastic container instance by adding the following configurations when you create resource objects for a Knative Service.
  - Specify the specification of the elastic container instance in the annotation parameter.
  - Specify virtual-kubelet in the nodeAffinity parameter.
  - Specify virtual-kubelet.io/provider in the tolerations parameter.

  Sample template:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
 name: helloworld-go
 namespace: vk
spec:
 template:
  metadata:
   annotations:
    k8s.aliyun.com/eci-use-specs: "2-4Gi"
  spec:
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
       - key: type
         operator: In
         values:
         - virtual-kubelet
   tolerations:
   - key: virtual-kubelet.io/provider
    operator: Exists
   containers:
   - env:
    - name: TARGET
      value: "Knative"
    image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
```

- If the virtual-node-affinity-injection=enabled label is added to the namespace, you need only to specify the specification of the elastic container instance in the annotation field to create an elastic container instance.

  ```
  kubectl label namespace vk virtual-node-affinity-injection=enabled
  ```

  Sample template:

  ```
  apiVersion: serving.knative.dev/v1
  kind: Service
  metadata:
   name: helloworld-go
   namespace: vk
  spec:
   template:
    metadata:
     annotations:
      k8s.aliyun.com/eci-use-specs: "2-4Gi"
    spec:
     containers:
     - env:
      - name: TARGET
        value: "Knative"
      image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
  ```

Method 2: Automatically create an elastic container instance for a Knative Service by specifying an ECS instance

You can automatically create an elastic container instance for a Knative Service by specifying an Elastic Compute Service (ECS) instance with specified labels in the nodeAffinity parameter. This way, the pod is deployed in both the ECS instance and the elastic container instance.

1. Add a label to the ECS instance. For more information, see Manage node labels.

   The *resource-role:ecs* label is added to the ECS instance in this example.

2. In the resource object template, set the nodeAffinity parameter.

   Specify the label that is added to the ECS instance and virtual-kubelet in the nodeAffinity parameter. Sample template:

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
 name: helloworld-go
spec:
 template:
  metadata:
   annotations:
    k8s.aliyun.com/eci-use-specs: "2-4Gi"
  spec:
   containerConcurrency: 10
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
       - key: resource-role
         operator: In
         values:
         - ecs
      - matchExpressions:
       - key: type
         operator: In
         values:
         - virtual-kubelet
   tolerations:
   - key: virtual-kubelet.io/provider
    operator: Exists
   containers:
   - env:
    - name: TARGET
      value: "Knative"
    image: registry.cn-hangzhou.aliyuncs.com/knative-sample/helloworld-go:73fbdd56
```

Result

Run the following command to verify the result. You can view that two elastic container instance-based pods are created.

```
kubectl -n vk get pod -o wide
```

Expected output:

```
NAME                                                READY   STATUS    RESTARTS   AGE   IP           NODE                 NOMINATED NODE   READINESS
GATES
helloworld-go-dqqhv-deployment-6d54c9c8dc-hkjwn     2/2     Running   0          40s   192.1xx.x.xx   virtual-node-eci-0   <none>           <none>
```

15.6.4. Use ARMS Prometheus to collect pod request-related metrics from a Knative Service

In practical scenarios, the number of concurrent pod requests is commonly used to evaluate a Service. Knative collects request-related metrics by using the queue-proxy container. This topic describes how to use Application Real-Time Monitoring Service (ARMS) Prometheus to collect pod request-related metrics from a Knative Service.

Step 1: Install the Prometheus Monitoring component

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab. Then, find and click **ack-arms-prometheus**.
4. On the **App Catalog - ack-arms-prometheus** page, select the cluster for which you want to enable ARMS Prometheus in the **Deploy** section, and click **Create**.

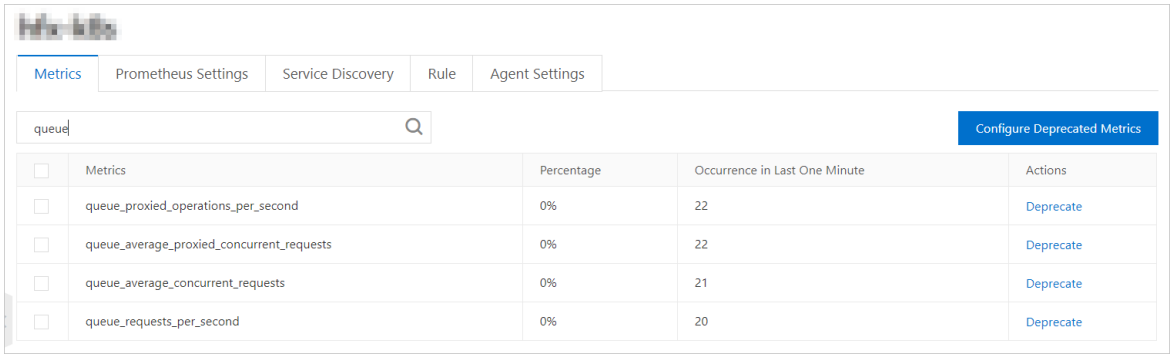
   > **Note** By default, **Namespace** and **Release Name** are set to **arms-prom**.

Step 2: Set collection job rules for the queue-proxy container

1. Log on to the ARMS console.
2. In the left-side navigation pane, click **Prometheus Monitoring**.
3. On the **Prometheus Monitoring** page, click **Settings** in the **Actions** column.
4. Click the **Prometheus Settings** tab, add the following content to the *Prometheus.yaml* file, and click **Save**.

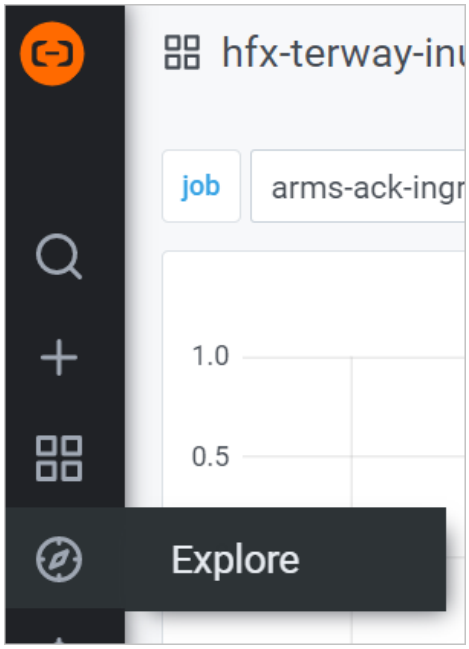
```
global:
  scrape_interval: 30s
  scrape_timeout: 10s
  evaluation_interval: 30s
scrape_configs:
- job_name: queue-proxy
  scrape_interval: 3s
  scrape_timeout: 3s
  kubernetes_sd_configs:
  - role: pod
  relabel_configs:
  # Rename metadata labels to be reader friendly
  - source_labels: [__meta_kubernetes_pod_label_serving_knative_dev_revision, __meta_kubernetes_pod_
container_port_name]
    action: keep
    regex: .+;http-autometric
  - source_labels: [__meta_kubernetes_namespace]
    target_label: namespace
  - source_labels: [__meta_kubernetes_pod_name]
    target_label: pod
  - source_labels: [__meta_kubernetes_service_name]
    target_label: service
```

5. Click the **Metrics** tab to view the metrics related to the queue-proxy.



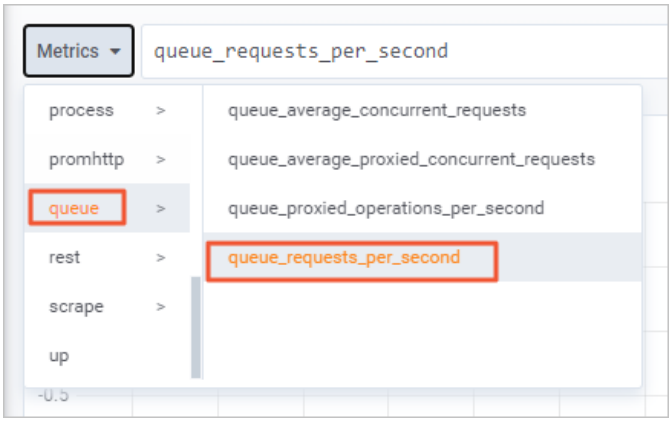
Step 3: Query the number of concurrent pod requests

1. In the left-side navigation pane of the cluster configuration page, click **Dashboards**.
2. On the **Dashboards** page, click **Prometheus** in the Name column.
3. On the left side of the **Prometheus** page, click the **Explore** icon.



4. In the **Metrics** drop-down list, select queue to view metrics related to queue-proxy.

   The **queue_requests_per_second** metric is collected in this example.



   You can also query the number of concurrent pod requests per second by specifying the namespace, revision, and pod name. The following code block is an example:

```
queue_requests_per_second{destination_configuration="helloworld-go",destination_namespace="default",destination_pod="helloworld-go-ttf52-deployment-5778d86bd6-dnxw2"}
```

15.6.5. Use HPA in Knative

You can use Horizontal Pod Autoscaler (HPA) in Knative to automatically scale pods. You can set the threshold of the CPU metric for a Knative Service. This ensures that pods are automatically scaled to match the fluctuations of user traffic. This topic describes how to use HPA in Knative.

Prerequisites

Deploy Knative

Procedure

1. Create the *ksvc-hpa.yaml* file.

   Configure an HPA scaling policy for a Knative Service. The following code block is an example:

   ```
   apiVersion: serving.knative.dev/v1
   kind: Service
   metadata:
    name: helloworld-go-hpa
   spec:
    template:
     metadata:
      labels:
       app: helloworld-go-hpa
      annotations:
       autoscaling.knative.dev/class: "hpa.autoscaling.knative.dev"
       autoscaling.knative.dev/metric: "cpu"
       autoscaling.knative.dev/target: "75"
       autoscaling.knative.dev/minScale: "1"
       autoscaling.knative.dev/maxScale: "10"
     spec:
      containers:
       - image: registry.cn-hangzhou.aliyuncs.com/knative-samples/helloworld-go:160e4dc8
         resources:
          requests:
           cpu: '200m'
   ```

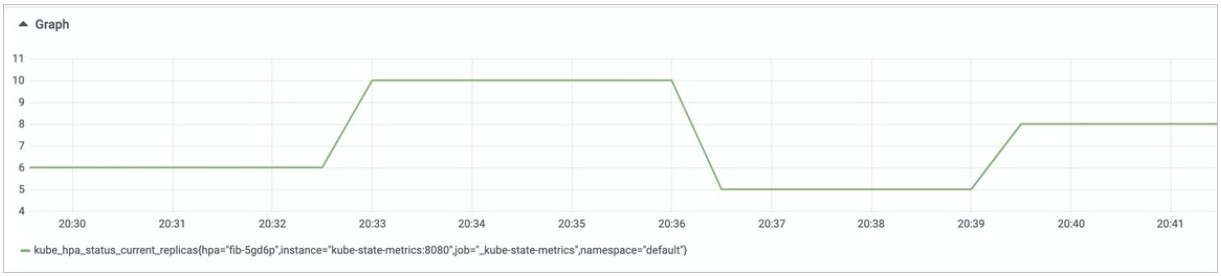
   - The `autoscaling.knative.dev/class: "hpa.autoscaling.knative.dev"` annotation specifies the HPA component for the Knative Service.
   - The `autoscaling.knative.dev/metric` annotation specifies the CPU metric.
   - The `autoscaling.knative.dev/target` annotation specifies the threshold of the CPU metric.
   - The `autoscaling.knative.dev/minScale: "1"` annotation specifies the minimum number of pods that must be guaranteed.
   - The `autoscaling.knative.dev/maxScale: "10"` annotation specifies the maximum number of pods that are allowed.

2. Apply the HPA scaling policy.

   ```
   kubectl apply -f ksvc-hpa.yaml
   ```

Result

Check the changes in the number of pods after HPA is enabled for the Knative Service. The following trend chart shows an example.

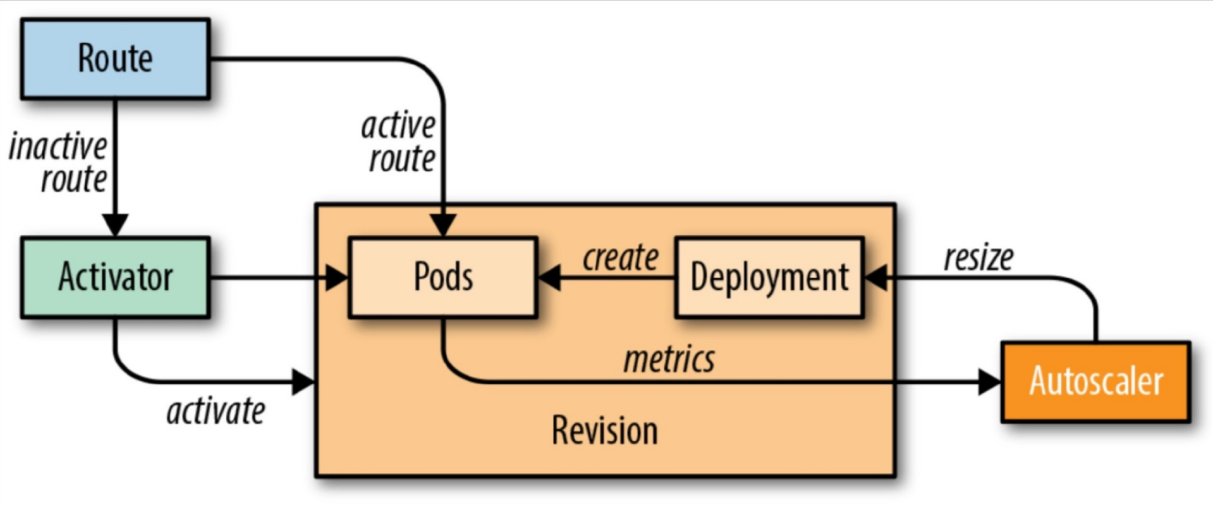


15.6.6. Enable automatic scaling for pods based on the number of requests

Knative Pod Autoscaler (KPA) is an out-of-the-box feature that can scale pods for your application to withstand traffic fluctuations. This topic describes how to enable auto scaling of pods based on the number of requests.

Context

Knative Serving adds the Queue-Proxy container to each pod. The Queue-Proxy container sends concurrency metrics of the application containers to KPA. After KPA receives the metrics, KPA automatically adjusts the number of pods provisioned for a Deployment based on the number of concurrent requests and related algorithms.



Concurrency and QPS

- The number of concurrent requests refers to the number of requests received by a pod at the same time.
- Queries per second (QPS) refers to the number of requests that can be handled by a pod per second. It also shows the maximum throughput of a pod.

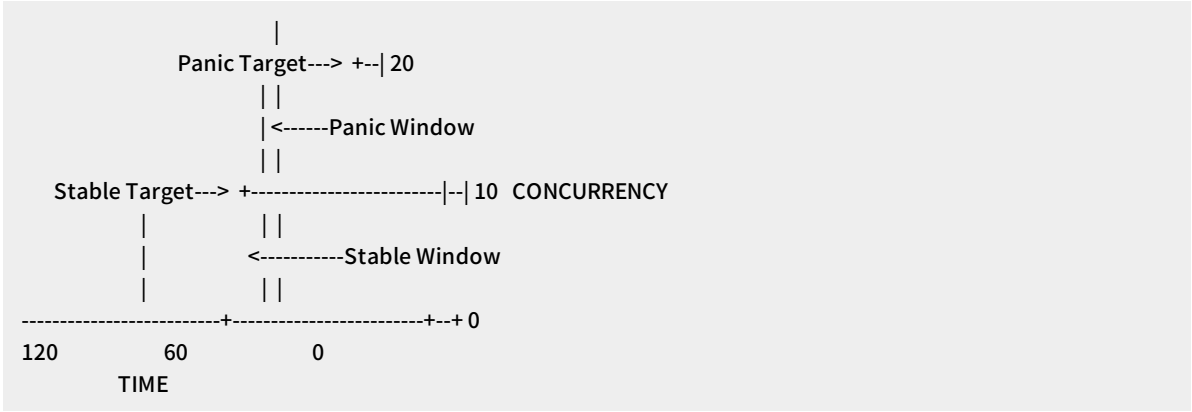
The increase of concurrent requests does not necessarily increase the QPS. In scenarios where the access load is heavy, if the concurrency is increased, the QPS may be decreased. This is because the system is overloaded and the CPU and memory usage is high, which downgrades the performance and increases the response latency.

Algorithms

KPA performs auto scaling based on the average number of concurrent requests per pod. This value is specified by the concurrency target. The default concurrency target is 100. The number of pods required to handle requests is determined based on the following formula: Number of pods = Total number of concurrent requests to the application/Concurrency target. For example, if you set the concurrency target to 10 and send 50 concurrent requests to an application, KPA creates five pods.

KPA provides two auto scaling modes: stable and panic.

- Stable
  In stable mode, KPA adjusts the number of pods provisioned for a Deployment to match the specified concurrency target. The concurrency target indicates the average number of requests received by a pod within a stable window of 60 seconds.
- Panic
  KPA calculates the average number of concurrent requests per pod within a stable window of 60 seconds. Therefore, the number of concurrent requests must remain at a specific level for 60 seconds. KPA also calculates the number of concurrent requests per pod within a panic window of 6 seconds. If the number of concurrent requests reaches twice the concurrency target, KAP switches to the panic mode. In panic mode, KPA scales pods within a shorter time window than in stable mode. After the burst of concurrent requests lasts for 60 seconds, KPA automatically switches back to the stable mode.



KPA configurations

- To configure KPA, you must configure config-autoscaler. By default, config-autoscaler is configured. The following content describes the key parameters.
  Run the following command to query config-autoscaler:

```
kubectl -n knative-serving get cm config-autoscaler
```

  Expected output:

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: config-autoscaler
 namespace: knative-serving
data:
 container-concurrency-target-default: "100"
 container-concurrency-target-percentage: "0.7"
 enable-scale-to-zero: "true"
 max-scale-up-rate: "1000"
 max-scale-down-rate: "2"
 panic-window-percentage: "10"
 panic-threshold-percentage: "200"
 scale-to-zero-grace-period: "30s"
 scale-to-zero-Pod-retention-period: "0s"
 stable-window: "60s"
 target-burst-capacity: "200"
 requests-per-second-target-default: "200"
```

- Configure parameters related to the scale-to-zero feature.
  - scale-to-zero-grace-period: The time period for which `inactive revison` keeps running before KPA scales the number of pods to zero. The minimum period is 30 seconds.

    ```
    scale-to-zero-grace-period: 30s
    ```

  - stable-window: In stable mode, KPA scales pods based on the average number of concurrent requests per pod within the stable window.

    ```
    stable-window: 60s
    ```

    You can also specify the stable window by using an annotation in the Deployment revision.

    ```
    autoscaling.knative.dev/window: 60s
    ```

  - enable-scale-to-zero: Set `enable-scale-to-zero` to `true`.

    ```
    enable-scale-to-zero: "true"
    ```

- Set a concurrency target for KPA.
  - target
    The target parameter sets a soft limit on the number of concurrent requests handled by each pod within a specified time period. We recommend that you use this parameter to control the concurrency. By default, the concurrency target in the ConfigMap is set to 100.

    ```
    `container-concurrency-target-default: 100`
    ```

    You can change the value by using the autoscaling.knative.dev/target per-revision annotation.

    ```
    autoscaling.knative.dev/target: 50
    ```

  - containerConcurrency
    The containerConcurrency parameter sets a hard limit on the number of concurrent requests handled by each pod within a specified time period. You can configure this parameter in the template section of the revision configuration file.

    > **Note**
    > To use containerConcurrency, make sure that the following conditions are met:
    > - Use containerConcurrency only when you want to limit the number of concurrent requests handled by an application within a specified time period.
    > - Use containerConcurrency only when you want to forcibly control the concurrency of an application.

    ```
    containerConcurrency: 0 | 1 | 2-N
    ```

    - 1: Only one request is handled at a time by each pod created based the current revision.
    - 2-N: Two or more concurrent requests are handled by each pod at a time.
    - 0: The number of concurrent requests handled by each pod is not limited. The number depends on the system.

  - container-concurrency-target-percentage
    This parameter is also known as a target utilization value or a concurrency factor. It is used in the calculation of the concurrency target for auto scaling. For example, the concurrency target or the containerConcurrency parameter is set to 100 and the target utilization value is set to 0.7. KPA adds pods to the application when the average number of concurrent requests per pod reaches 70 (100 × 0.7).
    The following formula applies: Concurrency value that triggers auto scaling = Concurrency target (or the value of containerConcurrency) × The value of container-concurrency-target-percentage.

- Set the scale bounds. You can use the minScale and maxScale parameters to set the scale bounds of pods that are provisioned for an application. You can use these two parameters to reduce cold starts and computing costs.

  > **Note**
  > - If you do not set the minScale annotation, all pods are removed when no traffic arrives. If you set enable-scale-to-zero to false in the config-autoscaler ConfigMap, KPA scales the number of pods to one when no traffic arrives.
  > - If you do not set the maxScale annotation, the number of pods that can be provisioned for the application is unlimited.

You can configure the `minScale` and `maxScale` parameters in the template section of the revision configuration file.

```
spec:
 template:
  metadata:
   autoscaling.knative.dev/minScale: "2"
   autoscaling.knative.dev/maxScale: "10"
```

Scenario 1: Enable auto scaling by setting a concurrency target

This example shows how to enable KPA to perform auto scaling by setting a concurrency target.

1. Use Knative to deploy serverless applications.
2. Create an *autoscale-go.yaml* file.

   Set the maximum number of concurrent requests per pod to 10.

   ```
   apiVersion: serving.knative.dev/v1alpha1
   kind: Service
   metadata:
    name: autoscale-go
    namespace: default
   spec:
    template:
     metadata:
      labels:
       app: autoscale-go
      annotations:
       autoscaling.knative.dev/target: "10"
     spec:
      containers:
       - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/autoscale-go:0.1
   ```

3. Run the following command to query the Kourier access gateway:

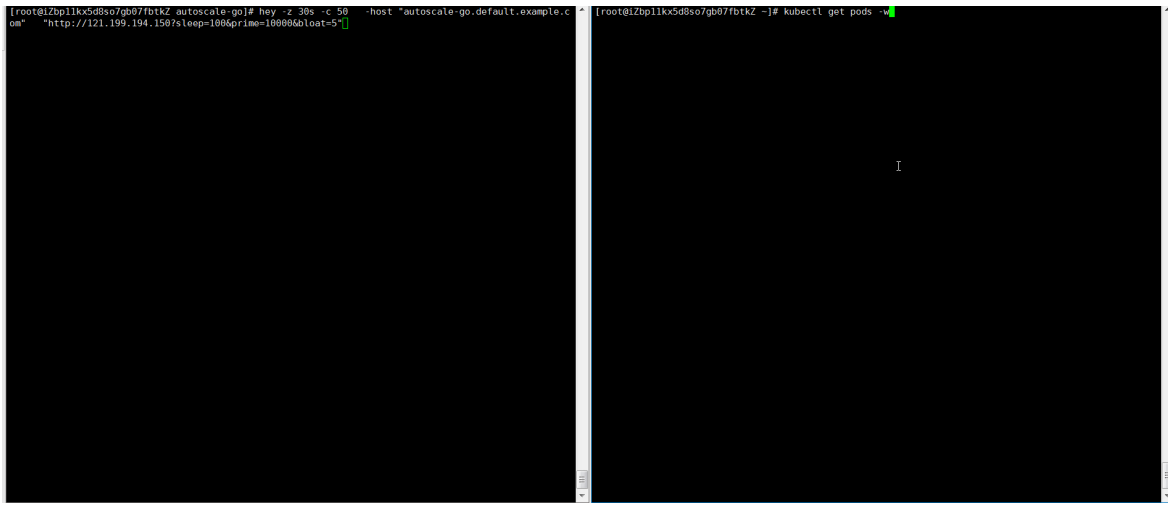
   ```
   kubectl -n knative-serving get svc kourier --output jsonpath="{.status.loadBalancer.ingress[*]['ip']}"
   ```

4. Use the load testing tool hey to send 50 concurrent requests to the application within 30 seconds.

   > **Note** For more information about hey, see hey.

   ```
   hey -z 30s -c 50  -host "autoscale-go.default.example.com"  "http://121.199.194.150?sleep=100&prime=10000
   &bloat=5"
   ```

   Expected output:



The output indicates that five pods are added.

Scenario 2: Enable auto scaling by setting scale bounds

Scale bounds control the minimum and maximum numbers of pods that can be provisioned for an application. This example shows how to enable auto scaling by setting scale bounds.

1. Use Knative to deploy serverless applications.
2. Create an *autoscale-go.yaml* file.

   Set the concurrency target to 10, `minScale` to 1, and `maxScale` to 3.

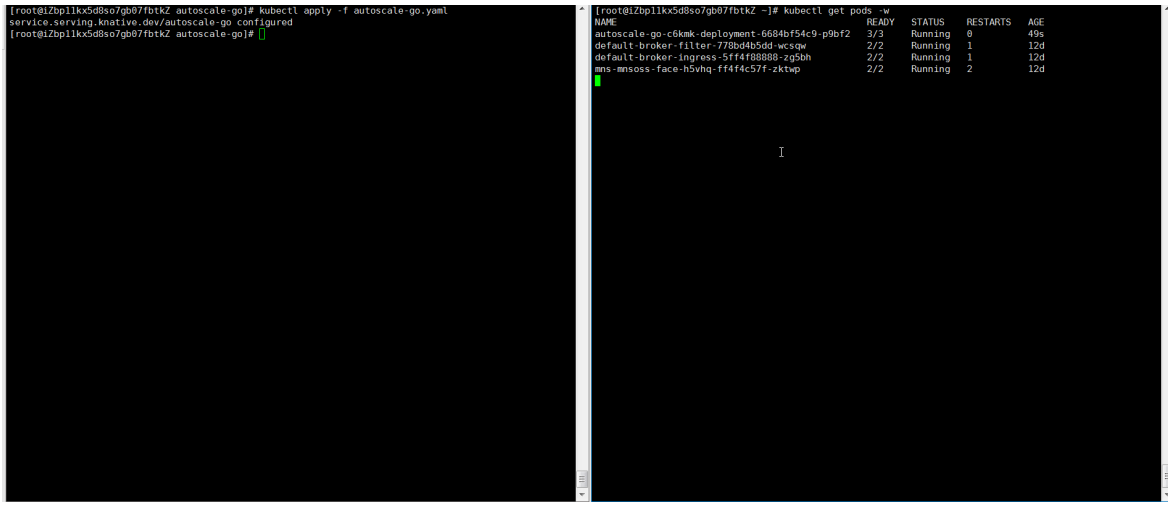
```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
 name: autoscale-go
 namespace: default
spec:
 template:
  metadata:
   labels:
    app: autoscale-go
   annotations:
    autoscaling.knative.dev/target: "10"
    autoscaling.knative.dev/minScale: "1"
    autoscaling.knative.dev/maxScale: "3"
  spec:
   containers:
    - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/autoscale-go:0.1
```

3. Use hey to send 50 concurrent requests to the application within 30 seconds.

   > **Note** For more information about hey, see hey.

```
hey -z 30s -c 50  -host "autoscale-go.default.example.com"  "http://121.199.194.150?sleep=100&prime=10000&bloat=5"
```

Expected output:



A maximum of three pods are added. One pod is reserved when no traffic flows to the application.

16.GPU/NPU

16.1. Create heterogeneous computing clusters

16.1.1. Create a managed Kubernetes cluster with GPU-accelerated nodes

This topic describes how to create a managed Kubernetes cluster for heterogeneous computing in the Container Service for Kubernetes (ACK) console.

Context

ACK performs the following operations when a cluster is created:

- Creates ECS instances, configures a public key to enable Secure Shell (SSH) logon from master nodes to other nodes, and then configures the ACK cluster through CloudInit.
- Creates a security group that allows access to the VPC over Internet Control Message Protocol (ICMP).
- If you do not specify an existing VPC, ACK creates a VPC and vSwitch and creates SNAT entries for the vSwitch.
- Adds route entries to the VPC.
- Creates a NAT gateway and EIPs.
- Creates a Resource Access Management (RAM) user and an AccessKey pair. Grants the following permissions to the RAM user: permissions to query, create, and delete ECS instances, permissions to add and delete disks, and full permissions on SLB, CloudMonitor, VPC, Log Service, and Apsara File Storage NAS. The ACK cluster automatically creates SLB instances, disks, and VPC route entries based on your configuration.
- Creates an internal-facing SLB instance and opens port 6443.
- Creates an Internet-facing SLB instance and opens ports 6443, 8443, and 22. If you enable SSH logon when you create the cluster, port 22 is opened. Otherwise, port 22 is not exposed.

Limits

- SLB instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
- ACK clusters support only VPCs.
- By default, each account has specific quotas on cloud resources that can be created. You cannot create clusters if the quota is reached. Make sure that you have sufficient quotas before you create a cluster. To request a quota increase, submit a ticket.
  - By default, you can create up to 50 clusters across all regions with each account. Each cluster can contain up to 100 nodes. To increase the quota of clusters or nodes,submit a ticket.

    > **Notice** By default, you can add up to 48 route entries to a VPC. This means that you can deploy up to 48 nodes in an ACK cluster that uses Flannel. An ACK cluster that uses Terway is not subject to this limit. To deploy more nodes in this case,submit a ticket to apply for an increase on the quota of route entries in the VPC that you want to use.

  - By default, you can create up to 100 security groups with each account.
  - By default, you can create up to 60 pay-as-you-go SLB instances with each account.

- By default, you can create up to 20 EIPs with each account.

- Limits on ECS instances:
  The pay-as-you-go and subscription billing methods are supported.

  > **Note** After an ECS instance is created, you can change its billing method from pay-as-you-go to subscription in the ECS console. For more information, see Change the billing method of an instance from pay-as-you-go to subscription.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Cluster Template**.
4. In the **Select Cluster Template** dialog box, find **Heterogeneous Computing Cluster** in the **Managed Clusters** section and click **Create**.
5. On the **Managed Kubernetes** tab, configure the cluster.
   i. Configure basic settings of the cluster.

Parameter	Description
Cluster Name	Enter a name for the cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
Cluster Specification	Select the required edition. By default, the Standard edition is selected.
Region	Select a region to deploy the cluster.
Resource Group	Move the pointer over **All Resources** at the top of the page and select the resource group to which the cluster belongs. The name of the selected resource group is displayed in the Resource Group field. 
Kubernetes Version	Select a Kubernetes version. The following versions are supported: 1.18.8-aliyun.1 and 1.16.9-aliyun.1.
Container Runtime	The **containerd**, **Docker**, and **Sandboxed-Container** runtimes are supported. For more information, see Comparison of Docker, containerd, and Sandboxed-Container.

Parameter	Description
VPC	Select a virtual private cloud (VPC) to deploy the cluster. Standard VPCs and shared VPCs are supported. ■ Shared VPC: The owner of a VPC (resource owner) can share the vSwitches in the VPC with other accounts in the same organization. ■ Standard VPC: The owner of a VPC (resource owner) cannot share the vSwitches in the VPC with other accounts. **Note** ACK clusters support only VPCs. You can select a VPC from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.
VSwitch	Set the vSwitch. You can select up to three vSwitches that are deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.
Network Plug-in	Select and configure the network plug-in. Flannel and Terway are available. For more information, see Use the Terway plug-in. ■ Flannel: an open source Container Network Interface (CNI) plug-in, which is simple and stable. Flannel provides a few simple features. However, it does not support standard Kubernetes network policies. ■ Terway: a network plug-in developed by Alibaba Cloud for ACK. Terway allows you to assign Alibaba Cloud elastic network interfaces (ENIs) to containers and use standard Kubernetes network policies to regulate how containers communicate with each other. Terway also supports bandwidth throttling on individual containers. **Note** ■ The number of pods that can be deployed on a node depends on the number of ENIs that are attached to the node and the maximum number of secondary IP addresses provided by these ENIs. ■ If you select a shared VPC for the cluster, you must select the Terway plug-in.

Parameter	Description
IP Addresses per Node	If you select **Flannel** as the network plug-in, you must set **IP Addresses per Node**. Note ■ **IP Addresses per Node** specifies the maximum number of IP addresses that can be assigned to each node. We recommend that you use the default value. ■ After you select the VPC and specify the number of IP addresses per node, recommended values are automatically generated for Pod CIDR block and Service CIDR block. The system also provides the maximum number of nodes that can be deployed in the cluster and the maximum number of pods that can be deployed on each node. You can modify the values based on your business requirements.
Pod CIDR Block	If you select **Flannel**, you must set the **Pod CIDR Block** parameter. The CIDR block specified by **Pod CIDR Block** cannot overlap with that of the VPC or the CIDR blocks of existing ACK clusters in the VPC. After you create the cluster, you cannot modify the pod CIDR block. In addition, the Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Terway Mode	If you set Network Plug-in to **Terway**, the **Terway Mode** parameter is available. When you set **Terway Mode**, you can select or clear **Assign One ENI to Each Pod**. ■ If you select **Assign One ENI to Each Pod**, an ENI is assigned to each pod. ■ If you clear **Assign One ENI to Each Pod**, an ENI is shared among multiple pods. A secondary IP address of the ENI is assigned to each pod. Note To use this feature, submit a ticket.

Parameter	Description
Service CIDR	Set the **Service CIDR** parameter. The CIDR block specified by **Service CIDR** cannot overlap with that of the VPC or the CIDR blocks of existing ACK clusters in the VPC. After you create the cluster, you cannot modify the Service CIDR block. In addition, the Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Configure SNAT	Specify whether to configure Source Network Address Translation (SNAT) rules for the VPC. ■ If the specified VPC has a Network Address Translation (NAT) gateway, ACK uses this NAT gateway. ■ Otherwise, the system automatically creates a NAT gateway. If you do not want the system to create a NAT gateway, clear **Configure SNAT for VPC**. In this case, you must manually create a NAT gateway and configure SNAT rules to enable Internet access for the VPC. Otherwise, ACK clusters in the VPC cannot access the Internet and the cluster cannot be created.
Access to API Server	By default, an internal-facing Server Load Balancer (SLB) instance is created for the cluster API server. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. **Notice** If you delete the SLB instance, you cannot access the cluster API server. Select or clear **Expose API Server with EIP**. The ACK API server provides multiple HTTP-based RESTful APIs, which can be used to create, delete, modify, query, and monitor resources, such as pods and Services. ■ If you select this check box, an elastic IP address (EIP) is created and associated with an Internet-facing SLB instance. Port 6443 used by the API server is opened on master nodes. You can connect to and manage the ACK cluster by using kubeconfig over the Internet. ■ If you clear this check box, no EIP is created. You can connect to and manage the ACK cluster by using kubeconfig only within the VPC.
RDS Whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the ACK cluster to the RDS whitelist.
Security Group	You can select **Create Basic Security Group**, **Create Advanced Security Group**, or **Select Existing Security Group**. For more information, see Overview.
Deletion Protection	

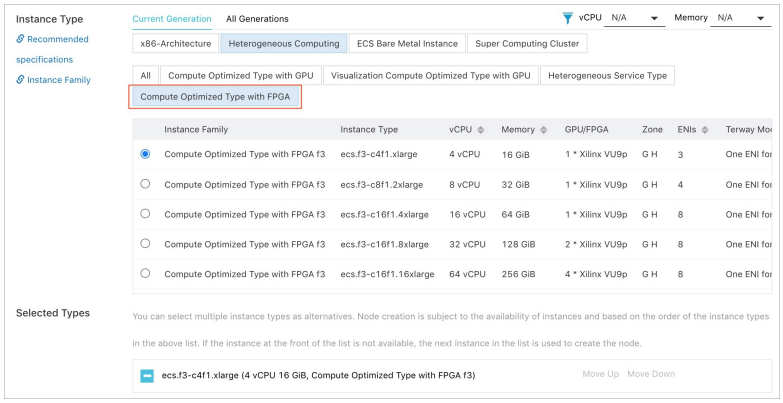
ii. Configure advanced settings.

Parameter	Description
Time Zone	Select a time zone for the ACK cluster. By default, the time zone configured for your browser is selected.

Parameter	Description
Kube-proxy Mode	iptables and IPVS are supported. ■ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the ACK cluster. This mode is suitable for ACK clusters that manage a small number of Services. ■ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for ACK clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.
Labels	Add labels to the cluster. Enter a key and a value, and click **Add**. **Note** ■ Key is required. *Value* is optional. ■ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ■ *Values* are not case-sensitive. A value can be empty and can contain up to 128 characters in length. It cannot be http:// or https://. ■ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the others that use the same key. ■ If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Cluster Domain	Set the domain name of the cluster. **Note** The default domain name is **cluster.local**. You can enter a custom domain name. A domain name consists of two parts. Each part must be 1 to 63 characters in length and can contain only letters and digits. You cannot leave these parts empty.
Custom Certificate SANs	You can enter custom subject alternative names (SANs) for the API server certificate of the cluster to accept requests from specified IP addresses or domain names. For more information, see Customize the SAN of the API server certificate for a managed Kubernetes cluster.
Service Account Token Volume Projection	**Service account token volume projection** reduces security risks when pods use service accounts to access the API server. This feature enables kubelet to request and store the token on behalf of the pod. This feature also allows you to configure token properties, such as the audience and validity duration. For more information, see Enable service account token volume projection.

Parameter	Description
Secret Encryption	If you select **Select Key**, you can use a key that is created in the Key Management Service (KMS) console to encrypt Kubernetes Secrets. For more information, see Use KMS to encrypt Kubernetes secrets at rest in the etcd.

6. Click **Next: Worker Configurations** to configure worker nodes.
   i. Set worker nodes.
   - If you select **Create Instance**, you must set the parameters that are listed in the following table.

Parameter	Description
Billing Method	These billing methods are supported: **Pay-As-You-Go** and **Subscription**.
Duration	If you select **Subscription**, you must specify the subscription duration. You can select 1 Month, 2 Months, 3 Months, or 6 Months. If you require a longer duration, you can select 1 Year, 2 Years, 3 Years, 4 Years, or 5 Years.
Auto Renewal	If you select **Subscription**, you must specify whether to enable **auto-renewal**.
Instance Type	Click **Heterogeneous Computing** and then click **Compute Optimized Type with GPU**. In the list of available instance types, select the instance types that you want to use. For more information, see Instance families. **Note** If no instance type is available, you can change vSwitches on the **Cluster Configurations** wizard page. 
Selected Types	The selected instance types are displayed.
Quantity	Specify the number of worker nodes that you want to create.

Parameter	Description
System Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. **Note** ■ You can select **Enable Backup** to back up disk data. ■ If you select **enhanced SSD** as the system disk type, you can set a custom **performance level** for the system disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.
Mount Data Disk	**ESSD Disk**, **SSD Disk**, and **Ultra Disk** are supported. You can select **Encrypt Disk** and **Enable Backup** when you mount data disks.
Operating System	ACK supports the following node operating systems: ■ Alibaba Cloud Linux 2. This is the default operating system. If you select Alibaba Cloud Linux 2, you can configure security reinforcement for the operating system: ■ Disable: disables security reinforcement for Alibaba Cloud Linux 2.x. ■ CIS Reinforcement: enables security reinforcement for Alibaba Cloud Linux 2.x. For more information about CIS reinforcement, see CIS reinforcement. ■ CentOS 7.x **Note** CentOS 8.x and later are not supported.
Logon Type Key Pair	■ Set the key pair. When you create the cluster, select a key pair that is used to log on to the cluster. If no key pair is available, click **create a key pair** to create one in the Elastic Compute Service (ECS) console. For more information, see Create an SSH key pair. After the key pair is created, set it as the credential to log on to the cluster. ■ Set the password. ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.

■ If you select **Add Existing Instance**, make sure that you have created ECS instances in the region where the cluster is deployed. Then, set **Operating System**, **Logon Type**, and **Key Pair** based on the preceding settings.

ii. Configure advanced settings of the worker nodes.

Parameter	Description
Node Protection	Specify whether to enable node protection. **Note** By default, this check box is selected. Node protection prevents nodes from being accidentally deleted in the console or by calling the API. This prevents user errors.
User Data	For more information, see Overview of ECS instance user data.
Custom Image	You can select a custom image for your ECS nodes. After you select a custom image, all nodes in the cluster are deployed by using this image. For more information about how to create a custom image, see Create a Kubernetes cluster by using a custom image. **Note** ■ Only custom images based on CentOS 7.x and Alibaba Cloud Linux 2.x are supported. ■ To use this feature, submit a ticket to apply to be added to a whitelist.
Custom Node Name	Specify whether to use a **custom node name**. A node name consists of a prefix, an IP substring, and a suffix. ■ Both the prefix and suffix can contain one or more parts that are separated by periods (.).These parts can contain lowercase letters, digits, and hyphens (-), and must start and end with a lowercase letter or digit. ■ The IP substring length specifies the number of digits to be truncated from the end of the returned node IP address. Valid values: 5 to 12. For example, if the node IP address is 192.1xx.x.xx, the prefix is aliyun.com, the IP substring length is 5, and the suffix is test, the node name will be aliyun.com00055test.
CPU Policy	Set the CPU policy. ■ none: This policy indicates that the default CPU affinity is used. This is the default policy. ■ static: This policy allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Taints	Add taints to the worker nodes in the ACK cluster.

7. Click **Next: Component Configurations** to configure components.

Parameter	Description

Parameter	Description
Ingress	Specify whether to install Ingress controllers. By default, **Install Ingress Controllers** is selected. For more information, see Configure an Ingress. **Note** If you want to select **Create Ingress Dashboard**, you must first enable Log Service.
Volume Plug-in	Select a volume plug-in. You can select Flexvolume or CSI. ACK clusters can be automatically bound to Alibaba Cloud disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets that are mounted to pods. For more information, see Storage management-FlexVolume and Storage management-CSI.
Monitoring Agents	Specify whether to install the CloudMonitor agent. By default, **Install CloudMonitor Agent on ECS Instance** and **Enable Prometheus Monitoring** are selected. After the CloudMonitor agent is installed on ECS nodes, you can view monitoring data about the nodes in the CloudMonitor console.
Alerts	Select **Use Default Alert Template** to enable the alerting feature and use the default alert rules. For more information, see Alert management.
Log Service	Specify whether to enable Log Service. You can select an existing Log Service project or create one. By default, **Enable Log Service** is selected. When you create an application, you can enable Log Service through a few steps. For more information, see Collect log files from containers by using Log Service. After you select **Enable Log Service**, you can specify whether to select **Create Ingress Dashboard** and **Install node-problem-detector and Create Event Center**.
Log Collection for Control Plane Components	If you select **Enable**, log of the control plane components is collected to the specified Log Service project that belongs to the current account. For more information, see Collect the logs of control plane components in a managed Kubernetes cluster.
Workflow Engine	Specify whether to enable Alibaba Cloud Genomics Service (AGS). **Note** To use this feature, submit a ticket to apply to be added to a whitelist. ○ If you select this check box, the system automatically installs the AGS workflow plug-in when the system creates the cluster. ○ If you clear this check box, you must manually install the AGS workflow plug-in. For more information, see Introduction to AGS CLI.

8. Click **Next: Confirm Order**.
9. Select **Terms of Service** and click **Create Cluster**.

Note It requires approximately 10 minutes for the system to create a managed Kubernetes cluster that contains multiple nodes.

What to do next

After the cluster is created, go to the **Clusters** page, find the created cluster, and then click the cluster name or click **Details** in the **Actions** column. In the left-side navigation pane of the details page, choose **Nodes > Nodes**. On the page that appears, select the worker node that is configured when you create the cluster, and choose **More > Details** in the **Actions** column. Then, you can view the GPU devices that are associated with the node.

16.1.2. Create a dedicated Kubernetes cluster with GPU-accelerated nodes

This topic describes how to create a dedicated Kubernetes cluster for heterogeneous computing in the Container Service for Kubernetes (ACK) console.

Context

ACK performs the following operations when a cluster is created:

- Creates ECS instances, configures a public key to enable Secure Shell (SSH) logon from master nodes to other nodes, and then configures the ACK cluster through CloudInit.
- Creates a security group that allows access to the VPC over Internet Control Message Protocol (ICMP).
- If you do not specify an existing VPC, ACK creates a VPC and vSwitch and creates SNAT entries for the vSwitch.
- Adds route entries to the VPC.
- Creates a NAT gateway and EIPs.
- Creates a Resource Access Management (RAM) user and an AccessKey pair. Grants the following permissions to the RAM user: permissions to query, create, and delete ECS instances, permissions to add and delete disks, and full permissions on SLB, CloudMonitor, VPC, Log Service, and Apsara File Storage NAS. The ACK cluster automatically creates SLB instances, disks, and VPC route entries based on your configuration.
- Creates an internal-facing SLB instance and opens port 6443.
- Creates an Internet-facing SLB instance and opens ports 6443, 8443, and 22. If you enable SSH logon when you create the cluster, port 22 is opened. Otherwise, port 22 is not exposed.

Limits

- SLB instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
- ACK clusters support only VPCs.
- By default, each account has specific quotas on cloud resources that can be created. You cannot create clusters if the quota is reached. Make sure that you have sufficient quotas before you create a cluster. To request a quota increase, submit a ticket.
  - By default, you can create up to 50 clusters across all regions with each account. Each cluster can contain up to 100 nodes. To increase the quota of clusters or nodes,submit a ticket.

    > **Notice** By default, you can add up to 48 route entries to a VPC. This means that you can deploy up to 48 nodes in an ACK cluster that uses Flannel. An ACK cluster that uses Terway is not subject to this limit. To deploy more nodes in this case,submit a ticket to apply for an increase on the quota of route entries in the VPC that you want to use.

  - By default, you can create up to 100 security groups with each account.
  - By default, you can create up to 60 pay-as-you-go SLB instances with each account.
  - By default, you can create up to 20 EIPs with each account.

- Limits on ECS instances:
  The pay-as-you-go and subscription billing methods are supported.

  > **Note** After an ECS instance is created, you can change its billing method from pay-as-you-go to subscription in the ECS console. For more information, see Change the billing method of an instance from pay-as-you-go to subscription.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Cluster Template**.
4. In the **Select Cluster Template** dialog box, find **Dedicated Cluster for Heterogeneous Computing** in the **Other Clusters** section and click **Create**.
5. On the **Dedicated Kubernetes** tab, configure the cluster.
   i. Configure basic settings of the cluster.

Parameter	Description
Cluster Name	Enter a name for the cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
Region	Select the region where you want to deploy the cluster.
Resource Group	Move the pointer over **All Resources** at the top of the page and select the resource group to which the cluster belongs. The name of the selected resource group is displayed in the Resource Group field. 
Kubernetes Version	Select a Kubernetes version.
Container Runtime	The **containerd**, **Docker**, and **Sandboxed-Container** runtimes are supported. For more information, see Comparison of Docker, containerd, and Sandboxed-Container.
VPC	Select a virtual private cloud (VPC) to deploy the cluster. Shared VPCs and standard VPCs are supported. ■ Shared VPC: The owner of a VPC (resource owner) can share vSwitches in the VPC with other accounts in the same organization. ■ Standard VPC: The owner of a VPC (resource owner) cannot share vSwitches in the VPC with other accounts. **Note** ACK clusters support only VPCs. You can select a VPC from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.

Parameter	Description
VSwitch	Set vSwitches. You can select up to three vSwitches that are deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.
Network Plug-in	Select and configure the network plug-in. Flannel and Terway are available. For more information, see Create a vSwitch. ■ Flannel: an open source Container Network Interface (CNI) plug-in, which is simple and stable. Flannel provides a few simple features and does not support standard Kubernetes network policies. ■ Terway: a network plug-in developed by Alibaba Cloud for ACK. Terway allows you to assign Alibaba Cloud elastic network interfaces (ENIs) to containers and use standard Kubernetes network policies to regulate how containers communicate with each other. Terway also supports bandwidth throttling on individual containers. **Note** ■ The number of pods that can be deployed on a node depends on the number of ENIs that are attached to the node and the maximum number of secondary IP addresses provided by these ENIs. ■ If you select a shared VPC for the cluster, you must select the Terway network plug-in.
Pod CIDR Block	If you select **Flannel**, you must set the **Pod CIDR Block** parameter. The CIDR block specified by **Pod CIDR Block** cannot overlap with that of the VPC or the CIDR blocks of existing ACK clusters in the VPC. After you create the cluster, you cannot modify the pod CIDR block. In addition, the Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Terway Mode	If you set Network Plug-in to **Terway**, the **Terway Mode** parameter is available. When you set **Terway Mode**, you can select or clear **Assign One ENI to Each Pod**. ■ If you select **Assign One ENI to Each Pod**, an ENI is assigned to each pod. ■ If you clear **Assign One ENI to Each Pod**, an ENI is shared among multiple pods. A secondary IP address of the ENI is assigned to each pod. **Note** This feature is available to only users in the whitelist. To apply to be added to the whitelist, submit a ticket.

Parameter	Description
Pod VSwitch	If you set Network Plug-in to **Terway**, the **Pod VSwitch** parameter is available. **Pod VSwitch** specifies vSwitches for pods. The ENIs that are assigned to pods must be in the same zone as the nodes that host the pods. For each vSwitch that is assigned to nodes, select a vSwitch for pods in the same zone. Pod vSwitches assign IP addresses to pods when the cluster is started. We recommend that you select vSwitches whose prefix length is no greater than 19 bits. This ensures a sufficient number of pods.
Service CIDR	Set the **Service CIDR** parameter. The CIDR block specified by **Service CIDR** cannot overlap with that of the VPC or the CIDR blocks of existing ACK clusters in the VPC. After you create the cluster, you cannot modify the Service CIDR block. In addition, the Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
IP Addresses per Node	If you select **Flannel**, you must specify a value for the **IP Addresses per Node** parameter. **Note** **IP Addresses per Node** specifies the maximum number of IP addresses that can be assigned to each node. We recommend that you use the default value.
Configure SNAT	Specify whether to configure Source Network Address Translation (SNAT) rules for the VPC. ■ If the specified VPC has a Network Address Translation (NAT) gateway, ACK uses this NAT gateway. ■ Otherwise, the system automatically creates a NAT gateway. If you do not want the system to create a NAT gateway, clear **Configure SNAT for VPC**. In this case, you must manually create a NAT gateway and configure SNAT rules to enable Internet access for the VPC. Otherwise, ACK clusters in the VPC cannot access the Internet and the cluster cannot be created.
Access to API Server	By default, an internal-facing Server Load Balancer (SLB) instance is created for the cluster API server. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. **Notice** If you delete the SLB instance, you cannot access the cluster API server. Select or clear **Expose API Server with EIP**. The ACK API server provides multiple HTTP-based RESTful APIs, which can be used to create, delete, modify, query, and monitor resources, such as pods and Services. ■ If you select this check box, an elastic IP address (EIP) is created and associated with an Internet-facing SLB instance. Port 6443 used by the API server is opened on master nodes. You can connect to and manage the ACK cluster by using kubeconfig over the Internet. ■ If you clear this check box, no EIP is created. You can connect to and manage the ACK cluster by using kubeconfig only within the VPC.

Parameter	Description
SSH Logon	To enable Secure Shell (SSH) logon, you must first select **Expose API Server with EIP**. ■ If you select Use SSH to Access the Cluster from the Internet, you can access the cluster through SSH. ■ If you clear Use SSH to Access the Cluster from the Internet, you cannot access the cluster through SSH or kubectl. If you want to access an Elastic Compute Service (ECS) instance in the cluster through SSH, you must manually bind an elastic IP address (EIP) to the ECS instance and configure security group rules to open SSH port 22. For more information, see Use SSH to connect to an ACK cluster.
RDS Whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the cluster to the RDS whitelist.
Security Group	You can select **Create Basic Security Group**, **Create Advanced Security Group**, or **Select Existing Security Group**. For more information, see Overview.

ii. Configure advanced settings.

Parameter	Description
Time Zone	Select a time zone for the ACK cluster. By default, the time zone configured for your browser is selected.
Kube-proxy Mode	iptables and IPVS are supported. ■ iptables is a kube-proxy mode. It uses iptables rules to conduct Service discovery and load balancing. The performance of this mode is limited by the size of the cluster. This mode is suitable for clusters that run a small number of Services. ■ IPVS is a high-performance kube-proxy mode. It uses Linux IP Virtual Server (IPVS) to conduct Service discovery and load balancing. This mode is suitable for clusters that run a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.

Parameter	Description
Labels	Add labels to the nodes in the cluster. Enter keys and values, and then click **Add**. **Note** ■ Key is required. *Value* is optional. ■ Keys cannot start with aliyun, http://, or https://. Keys are not case-sensitive and cannot exceed 64 characters in length. ■ *Values* are not case-sensitive. A value must not exceed 128 characters in length, and cannot start with http:// or https://. ■ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the one that uses the same key. ■ You can add up to 20 labels to each resource. If you add more than 20 labels to a resource, all labels become invalid. You must remove unused labels for the other labels to take effect.
Custom Image	You can select a custom image to replace the default image.
Cluster Domain	Set the domain name of the cluster. **Note** The default domain name is **cluster.local**. You can enter a custom domain name. A domain name consists of two parts. Each part must not exceed 63 characters in length, and can contain only letters and digits. You cannot leave these parts empty.
Custom Certificate SANs	You can specify custom subject alternative names (SANs) for the API server certificate. Separate multiple IP addresses or domain names with commas (,).
Service Account Token Volume Projection	Enable **Service Account Token Volume Projection** to enhance security when you use service accounts. For more information, see Enable service account token volume projection.
Cluster CA	If you select Custom Cluster CA, upload a Certificate Authority (CA) certificate for the cluster to ensure secure data transmission between the server and client.
Deletion Protection	Specify whether to enable deletion protection. If you select this check box, the cluster cannot be deleted in the console or by calling API operations. This protects the cluster from being accidentally deleted.

6. Click **Next: Master Configurations** to configure master nodes.

Parameter	Description
Billing Method	**Pay-As-You-Go** and **Subscription** are supported.

Parameter	Description
Duration	If you select **Subscription**, you must specify the subscription duration.
Auto Renewal	If you select **Subscription**, you must specify whether to enable **Auto Renewal**.
Master Node Quantity	Specify the number of master nodes. You can create three or five master nodes.
Instance Type	Select an instance type for the master nodes. For more information, see Instance families. **Note** If no instance type is available, you can change vSwitches on the **Cluster Configurations** wizard page.
System Disk	By default, system disks are mounted to master nodes. **ESSD Disk**, **SSD Disk**, and **Ultra Disk** are supported. **Note** ◦ You can select **Enable Backup** to back up disk data. ◦ If you select an **enhanced SSD** as the system disk, you can set a **performance level** for the disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.

7. Click **Next: Worker Configurations** to configure worker nodes.
   i. Set worker instances.
   - If you select **Create Instance**, you must set the parameters that are listed in the following table.

Parameter	Description
Instance Type	Choose **Heterogeneous Computing GPU/FPGA/NPU** > **Compute Optimized Type with GPU**. In the list of available instance types, select one or more required instance types. For more information, see Instance families. **Note** If no instance type is available, you can change vSwitches on the **Cluster Configurations** wizard page.
Selected Types	The selected instance types are displayed.
Quantity	Specify the number of worker nodes.

Parameter	Description
System Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. **Note** ■ You can select **Enable Backup** to back up disk data. ■ If you select **enhanced SSD** as the system disk type, you can set a custom **performance level** for the system disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.
Mount Data Disk	**ESSD Disk**, **SSD Disk**, and **Ultra Disk** are supported. You can select **Encrypt Disk** and **Enable Backup** when you mount data disks.
Operating System	ACK supports the following node operating systems: ■ Alibaba Cloud Linux 2. This is the default operating system. If you select Alibaba Cloud Linux 2, you can configure security reinforcement for the operating system: ■ Disable: disables security reinforcement for Alibaba Cloud Linux 2.x. ■ CIS Reinforcement: enables security reinforcement for Alibaba Cloud Linux 2.x. For more information about CIS reinforcement, see CIS reinforcement. ■ CentOS 7.x **Note** CentOS 8.x and later are not supported.
Logon Type	■ Key pair logon ■ **Key Pair**: Select an SSH key pair from the drop-down list. ■ **create a key pair**: Create an SSH key pair if none is available. For more information about how to create an SSH key pair, see Create an SSH key pair. After the key pair is created, set it as the credential that is used to log on to the cluster. ■ Password logon ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.

- If you select **Add Existing Instance**, make sure that you have created ECS instances in the region where the cluster is deployed. Then, set **Operating System**, **Logon Type**, and **Key Pair** based on the preceding settings.

ii. Configure advanced settings.

Parameter	Description
Node Protection	Specify whether to enable node protection. **Note** By default, this check box is selected. The nodes in the cluster cannot be deleted in the console or by calling the API. This protects the nodes from being accidentally deleted.
User Data	For more information, see Overview of ECS instance user data.
Custom Node Name	Specify whether to enable **Custom Node Name**. A node name consists of a prefix, an IP substring, and a suffix. ■ Both the prefix and suffix can contain one or more parts that are separated with periods (.).These parts can contain lowercase letters, digits, and hyphens (-), and must start and end with a digit or lowercase letter. ■ The IP substring length specifies the number of digits to be truncated from the end of the node IP address. Valid values: 5 to 12. For example, if the node IP address is 192.168.0.55, the prefix is aliyun.com, the IP substring length is 5, and the suffix is test, the node name is aliyun.com00055test.
Node Port Range	Set the node port range. The default port range is 30000 to 32767.
CPU Policy	Set the CPU policy. ■ None: indicates that the default CPU affinity is used. This is the default policy. ■ Static: allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Taints	Add taints to all worker nodes in the cluster.

8. Click **Next:Component Configurations** to configure components.

Parameter	Description
Ingress	Specify whether to install Ingress controllers. By default, **Install Ingress Controllers** is selected. For more information, see Ingress高级用法.
Volume Plug-in	Select a volume plug-in. FlexVolume and CSI are supported. ACK clusters can be automatically bound to Alibaba Cloud disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets that are mounted to pods. For more information, see Storage management-Flexvolume and Storage management-CSI.
Monitoring Agents	Specify whether to install the CloudMonitor agent. You can select **Install CloudMonitor Agent on ECS Instance**. After the CloudMonitor agent is installed, you can view monitoring information about ECS instances in the CloudMonitor console.

Parameter	Description
Log Service	Specify whether to enable Log Service. You can select an existing project or create a project. If you select **Enable Log Service**, the Log Service plug-in is automatically installed in the cluster. When you deploy an application, you can activate Log Service by performing a few steps. For more information, see Collect log files from containers by using Log Service. After you select **Enable Log Service**, you can specify whether to select **Create Ingress Dashboard** and **Install node-problem-detector and Create Event Center**.
Workflow Engine	Specify whether to enable Alibaba Cloud Genomics Compute Service (AGS). **Note** This feature is available to only users in the whitelist. ○ If you select this check box, the system automatically installs the AGS workflow plug-in when the system creates the cluster. ○ If you clear this check box, you must manually install the AGS workflow plug-in. For more information, see Introduction to AGS CLI.

9. Click **Next: Confirm Order**.
10. Select **Terms of Service** and click **Create Cluster**.

Note It requires approximately 10 minutes for the system to create a managed Kubernetes cluster that contains multiple nodes.

What to do next

After the cluster is created, go to the **Clusters** page, find the created cluster, and then click the cluster name or click **Details** in the **Actions** column. In the left-side navigation pane of the details page, choose **Nodes > Nodes**. On the page that appears, select the worker node that is configured when you create the cluster, and choose **More > Details** in the **Actions** column. Then, you can view the GPU devices that are associated with the node.

16.1.3. Create a Kubernetes cluster with NPU-accelerated nodes

This topic describes how to create a Kubernetes cluster with NPU-accelerated nodes in the Container Service for Kubernetes (ACK) console.

Context

ACK performs the following operations when a cluster is created:

- Creates ECS instances, configures a public key to enable Secure Shell (SSH) logon from master nodes to other nodes, and then configures the ACK cluster through CloudInit.
- Creates a security group that allows access to the VPC over Internet Control Message Protocol (ICMP).
- If you do not specify an existing VPC, ACK creates a VPC and vSwitch and creates SNAT entries for the vSwitch.
- Adds route entries to the VPC.
- Creates a NAT gateway and EIPs.

- Creates a Resource Access Management (RAM) user and an AccessKey pair. Grants the following permissions to the RAM user: permissions to query, create, and delete ECS instances, permissions to add and delete disks, and full permissions on SLB, CloudMonitor, VPC, Log Service, and Apsara File Storage NAS. The ACK cluster automatically creates SLB instances, disks, and VPC route entries based on your configuration.
- Creates an internal-facing SLB instance and opens port 6443.
- Creates an Internet-facing SLB instance and opens ports 6443, 8443, and 22. If you enable SSH logon when you create the cluster, port 22 is opened. Otherwise, port 22 is not exposed.

Limits

- SLB instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
- ACK clusters support only VPCs.
- By default, each account has specific quotas on cloud resources that can be created. You cannot create clusters if the quota is reached. Make sure that you have sufficient quotas before you create a cluster. To request a quota increase, submit a ticket.
  - By default, you can create up to 50 clusters across all regions with each account. Each cluster can contain up to 100 nodes. To increase the quota of clusters or nodes,submit a ticket.

    > **Notice** By default, you can add up to 48 route entries to a VPC. This means that you can deploy up to 48 nodes in an ACK cluster that uses Flannel. An ACK cluster that uses Terway is not subject to this limit. To deploy more nodes in this case,submit a ticket to apply for an increase on the quota of route entries in the VPC that you want to use.

  - By default, you can create up to 100 security groups with each account.
  - By default, you can create up to 60 pay-as-you-go SLB instances with each account.
  - By default, you can create up to 20 EIPs with each account.
- Limits on ECS instances:
  The pay-as-you-go and subscription billing methods are supported.

  > **Note** After an ECS instance is created, you can change its billing method from pay-as-you-go to subscription in the ECS console. For more information, see Change the billing method of an instance from pay-as-you-go to subscription.

Create a Kubernetes cluster with NPU-accelerated nodes

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Cluster Template**.
4. On the **Select Cluster Template** page, find **Dedicated Cluster for Heterogeneous Computing** and click **Create** to go to the Cluster Configurations wizard page.

   In this example, a dedicated Kubernetes cluster for heterogeneous computing is created. You can also select **Heterogeneous Computing Cluster** on the Select Cluster Template page to create a managed Kubernetes cluster for heterogeneous computing.

   > **Note** To create a Kubernetes cluster with NPU-accelerated nodes, you must select NPU-accelerated Elastic Compute Service (ECS) instances as worker nodes. For more information about other parameters, see 创建Kubernetes托管版集群.

5. Configure worker nodes. In this example, NPU-accelerated nodes are used as worker nodes and the

ecs.ebman1.26xlarg instance type is selected.

- To create new instances, you must specify the instance family, instance type, and the number of worker nodes that you want to create. In this example, two NPU-accelerated nodes are created and the instance type is ecs.ebman1.26xlarge.
- To add existing instances, you must first create NPU-accelerated instances in the region where you want to create the cluster. For more information, see Instance families.

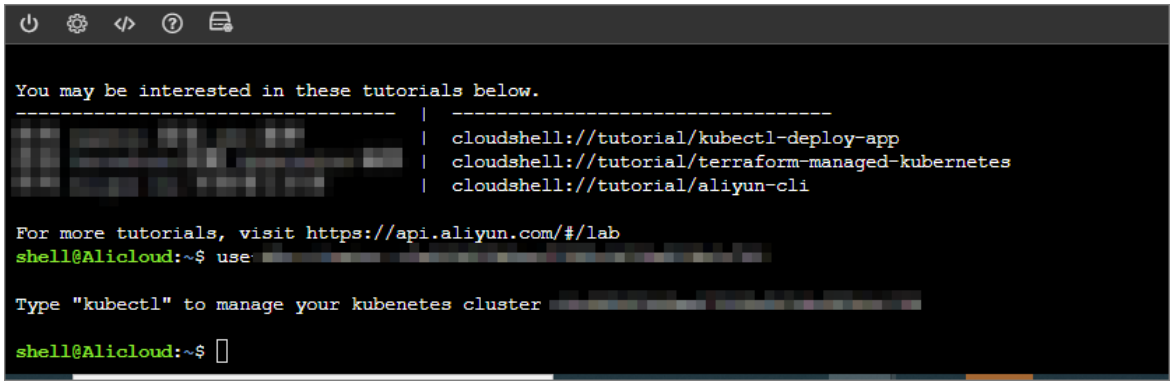
6. Set the other parameters and click **Create Cluster** to start the deployment.
7. Check the NPU devices that are mounted to the nodes.
   i. In the left-side navigation pane of the ACK console, click **Clusters**.
   ii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster, or click **Applications** in the **Actions** column.
   iii. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
   iv. On the Nodes page, find the node that you created and choose **More > Details** in the **Actions** column. On the details page of the node, you can view the NPU device that is mounted to the node.

Configure a Secret to pull private images

Before you can use Docker images with NPU capabilities that are provided by Alibaba Cloud, contact your product managers to obtain an authorized Docker registry account. Download the required Docker image and configure a Secret to pull private images in the cluster.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and choose **More > Open Cloud Shell** in the **Actions** column.

   After you connect to the cluster, the following output is returned:



4. Run the following commands to create a docker-registry Secret:

```
kubectl create secret \
docker-registry regsecret \
--docker-server=registry.cn-shanghai.aliyuncs.com \
--docker-username=<your_username>
--docker-password=<your_password>
```

> **Note**
> - `regsecret`: the name of the Secret. You can enter a custom name.
> - `--docker-server`: the address of the Docker registry.
> - `--docker-username`: the username of the Docker registry account.
> - `--docker-password`: the password of the Docker registry account.

5. Add the Secret in the pod configuration file to pull the private image with NPU capabilities.

```
apiVersion: v1
kind: Pod
metadata:
 name: test-npu
spec:
 containers:
 - name: <The name of the container>
  image: registry.cn-shanghai.aliyuncs.com/hgai/<The Docker image with NPU capabilities>
 imagePullSecrets:
 - name: <The name of the Secret>
```

> **Note**
> - `imagePullSecrets` specifies the Secret that is required to pull the image.
> - `regsecret` must be the same as the name of the Secret that is created in Step .
> - The Docker registry address specified by `image` must be the same as the one that is specified by `--docker-server`.

Use a Kubernetes cluster with NPU-accelerated nodes

You can manage ALI NPU devices in Kubernetes clusters. This allows you to deploy AI inference tasks with a few clicks and monitor the usage of the NPU resources. For more information, see Perform NPU scheduling in a Kubernetes cluster.

16.1.4. Create a managed Kubernetes cluster with FPGA-acceleated nodes

This topic describes how to create a managed Kubernetes cluster with field-programmable gate array (FPGA)-accelerated nodes in the Container Service for Kubernetes (ACK) console.

Prerequisites

- Select the region where you want to create a managed Kubernetes cluster with FPGA-accelerated nodes. Only specific regions and zones support FPGA-accelerated Elastic Compute Service (ECS) instances. For more information, see ECS instance types available for each region.
- FPGA-accelerated ECS instances must use an image that provides the Xilinx development environment. This image can be only shared to users by the FPGA as a service (FaaS) team.Submit a ticket to the FaaS team to use this image.
- Clone a custom image named *faas_f3* from the shared image. For more information, see Copy custom images.

Context

ACK performs the following operations when a cluster is created:

- Creates ECS instances, configures a public key to enable Secure Shell (SSH) logon from master nodes to other nodes, and then configures the ACK cluster through CloudInit.
- Creates a security group that allows access to the VPC over Internet Control Message Protocol (ICMP).
- If you do not specify an existing VPC, ACK creates a VPC and vSwitch and creates SNAT entries for the vSwitch.
- Adds route entries to the VPC.
- Creates a NAT gateway and EIPs.
- Creates a Resource Access Management (RAM) user and an AccessKey pair. Grants the following permissions to the RAM user: permissions to query, create, and delete ECS instances, permissions to add and delete disks, and full permissions on SLB, CloudMonitor, VPC, Log Service, and Apsara File Storage NAS. The ACK cluster automatically creates SLB instances, disks, and VPC route entries based on your configuration.
- Creates an internal-facing SLB instance and opens port 6443.
- Creates an Internet-facing SLB instance and opens ports 6443, 8443, and 22. If you enable SSH logon when you create the cluster, port 22 is opened. Otherwise, port 22 is not exposed.

Limits

- SLB instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
- ACK clusters support only VPCs.
- By default, each account has specific quotas on cloud resources that can be created. You cannot create clusters if the quota is reached. Make sure that you have sufficient quotas before you create a cluster. To request a quota increase, submit a ticket.
  - By default, you can create up to 50 clusters across all regions with each account. Each cluster can contain up to 100 nodes. To increase the quota of clusters or nodes,submit a ticket.

    > **Notice** By default, you can add up to 48 route entries to a VPC. This means that you can deploy up to 48 nodes in an ACK cluster that uses Flannel. An ACK cluster that uses Terway is not subject to this limit. To deploy more nodes in this case,submit a ticket to apply for an increase on the quota of route entries in the VPC that you want to use.

  - By default, you can create up to 100 security groups with each account.
  - By default, you can create up to 60 pay-as-you-go SLB instances with each account.
  - By default, you can create up to 20 EIPs with each account.
- Limits on ECS instances:
  The pay-as-you-go and subscription billing methods are supported.

  > **Note** After an ECS instance is created, you can change its billing method from pay-as-you-go to subscription in the ECS console. For more information, see Change the billing method of an instance from pay-as-you-go to subscription.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Cluster Template**.
4. In the **Select Cluster Template** dialog box, find **Heterogeneous Computing Cluster** in the

Managed Clusters section and click **Create**.

5. On the **Managed Kubernetes** tab, configure the cluster.
   i. Configure basic settings of the cluster.

Parameter	Description
Cluster Name	Enter a name for the cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
Cluster Specification	Select the required edition. By default, the Standard edition is selected.
Region	Select the region where you want to deploy the cluster.
Resource Group	Move the pointer over **All Resources** at the top of the page and select the resource group to which the cluster belongs. The name of the selected resource group is displayed in the Resource Group field. 
Kubernetes Version	Select a Kubernetes version. The following versions are supported: 1.18.8-aliyun.1 and 1.16.9-aliyun.1.
Container Runtime	The **containerd**, **Docker**, and **Sandboxed-Container** runtimes are supported. For more information, see Comparison of Docker, containerd, and Sandboxed-Container.
VPC	Select a virtual private cloud (VPC) to deploy the cluster. Shared VPCs and standard VPCs are supported. ■ Shared VPC: The owner of a VPC (resource owner) can share vSwitches in the VPC with other accounts in the same organization. ■ Standard VPC: The owner of a VPC (resource owner) cannot share vSwitches in the VPC with other accounts. **Note** ACK clusters support only VPCs. You can select a VPC from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.
VSwitch	Set the vSwitch. You can select up to three vSwitches that are deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.

Parameter	Description
Network Plug-in	Select and configure the network plug-in. Flannel and Terway are available. For more information, see Use the Terway plug-in. ■ Flannel: an open source Container Network Interface (CNI) plug-in, which is simple and stable. Flannel provides a few simple features. However, it does not support standard Kubernetes network policies. ■ Terway: a network plug-in developed by Alibaba Cloud for ACK. Terway allows you to assign Alibaba Cloud elastic network interfaces (ENIs) to containers and use standard Kubernetes network policies to regulate how containers communicate with each other. Terway also supports bandwidth throttling on individual containers. **Note** ■ The number of pods that can be deployed on a node depends on the number of ENIs that are attached to the node and the maximum number of secondary IP addresses provided by these ENIs. ■ If you select a shared VPC for the cluster, you must select the Terway plug-in.
Pod CIDR Block	If you select **Flannel**, you must set the **Pod CIDR Block** parameter. The CIDR block specified by **Pod CIDR Block** cannot overlap with that of the VPC or the CIDR blocks of existing ACK clusters in the VPC. After you create the cluster, you cannot modify the pod CIDR block. In addition, the Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Terway Mode	If you select **Terway**, the **Terway Mode** parameter is available. When you set **Terway Mode**, you can select or clear **Assign One ENI to Each Pod**. ■ If you select **Assign One ENI to Each Pod**, an ENI is assigned to each pod. ■ If you clear **Assign One ENI to Each Pod**, an ENI is shared among multiple pods. A secondary IP address of the ENI is assigned to each pod. **Note** This feature is available to only users in the whitelist. To apply to be added to the whitelist, submit a ticket.
Service CIDR	Set the **Service CIDR** parameter. The CIDR block specified by **Service CIDR** cannot overlap with that of the VPC or the CIDR blocks of existing ACK clusters in the VPC. After you create the cluster, you cannot modify the Service CIDR block. In addition, the Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.

Parameter	Description
IP Addresses per Node	If you select **Flannel**, you must specify a value for the **IP Addresses per Node** parameter. **Note** **IP Addresses per Node** specifies the maximum number of IP addresses that can be assigned to each node. We recommend that you use the default value.
Configure SNAT	Specify whether to configure Source Network Address Translation (SNAT) rules for the VPC. ■ If the specified VPC has a Network Address Translation (NAT) gateway, ACK uses this NAT gateway. ■ Otherwise, the system automatically creates a NAT gateway. If you do not want the system to create a NAT gateway, clear **Configure SNAT for VPC**. In this case, you must manually create a NAT gateway and configure SNAT rules to enable Internet access for the VPC. Otherwise, ACK clusters in the VPC cannot access the Internet and the cluster cannot be created.
Access to API Server	By default, an internal-facing Server Load Balancer (SLB) instance is created for the cluster API server. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. **Notice** If you delete the SLB instance, you cannot access the cluster API server. Select or clear **Expose API Server with EIP**. The ACK API server provides multiple HTTP-based RESTful APIs, which can be used to create, delete, modify, query, and monitor resources, such as pods and Services. ■ If you select this check box, an elastic IP address (EIP) is created and associated with an Internet-facing SLB instance. Port 6443 used by the API server is opened on master nodes. You can connect to and manage the ACK cluster by using kubeconfig over the Internet. ■ If you clear this check box, no EIP is created. You can connect to and manage the ACK cluster by using kubeconfig only within the VPC.
RDS Whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the cluster to the RDS whitelist.
Security Group	You can select **Create Basic Security Group**, **Create Advanced Security Group**, or **Select Existing Security Group**. For more information, see Overview.

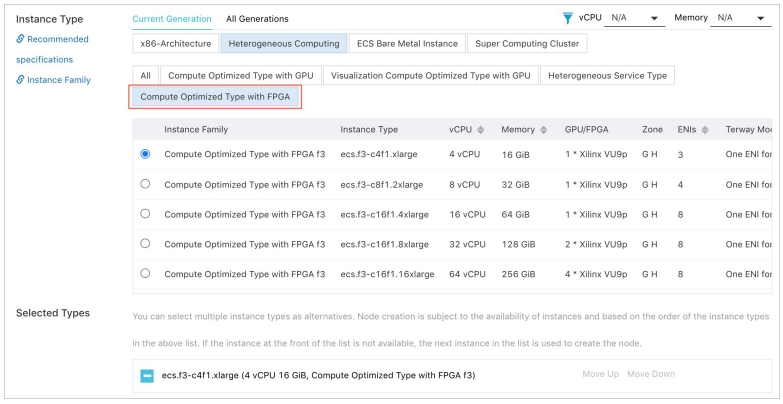
ii. Configure advanced settings of the cluster.

Parameter	Description
Time Zone	Select a time zone for the ACK cluster. By default, the time zone configured for your browser is selected.

Parameter	Description
Kube-proxy Mode	iptables and IPVS are supported. ■ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the ACK cluster. This mode is suitable for ACK clusters that manage a small number of Services. ■ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for ACK clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.
Labels	Add labels to the cluster. Enter a key and a value, and click **Add**. **Note** ■ Key is required. *Value* is optional. ■ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ■ *Values* are not case-sensitive. A value can be empty and can contain up to 128 characters in length. It cannot be http:// or https://. ■ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the others that use the same key. ■ If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Cluster Domain	Set the domain name of the cluster. **Note** The default domain name is **cluster.local**. You can enter a custom domain name. A domain name consists of two parts. Each part must be 1 to 63 characters in length and can contain only letters and digits. You cannot leave these parts empty.
Custom Certificate SANs	You can enter custom subject alternative names (SANs) for the API server certificate of the cluster to accept requests from specified IP addresses or domain names. For more information, see Customize the SAN of the API server certificate for a managed Kubernetes cluster.
Service Account Token Volume Projection	**Service account token volume projection** reduces security risks when pods use service accounts to access the API server. This feature enables kubelet to request and store the token on behalf of the pod. This feature also allows you to configure token properties, such as the audience and validity duration. For more information, see Enable service account token volume projection.

Parameter	Description
Deletion Protection	

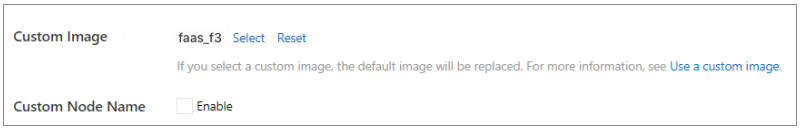
6. Click **Next: Worker Configurations** to configure worker nodes.
   i. Select ECS instances for worker nodes.
   - If you select **Create Instance**, you must set the parameters that are described in the following table.

Parameter	Description
Billing Method	The following billing methods are supported: **Pay-As-You-Go** and **Subscription**.
Duration	If you select **Subscription**, you must specify the subscription duration. You can select 1, 2, 3, or 6 months, or 1 to 5 years.
Auto Renewal	If you select **Subscription**, you must specify whether to enable **Auto Renewal**.
Instance Type	Choose **Heterogeneous Computing GPU/FPGA/NPU** > **Compute Optimized Type with FPGA**. In the list of available instance types, select one or more required instance types. For more information, see Instance families. **Note** If no instance type is available, you can change the vSwitch on the **Cluster Configurations** wizard page. 
Selected Types	The selected ECS instance types are displayed.
Quantity	Specify the number of worker nodes (ECS instances) to be created.

Parameter	Description
System Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. **Note** ■ You can select **Enable Backup** to back up disk data. ■ If you select **enhanced SSD** as the system disk type, you can set a custom **performance level** for the system disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.
Mount Data Disk	**ESSD Disk**, **SSD Disk**, and **Ultra Disk** are supported. You can select **Encrypt Disk** and **Enable Backup** when you mount data disks.
Operating System	ACK supports the following node operating systems: ■ Alibaba Cloud Linux 2. This is the default operating system. If you select Alibaba Cloud Linux 2, you can configure security reinforcement for the operating system: ■ Disable: disables security reinforcement for Alibaba Cloud Linux 2.x. ■ CIS Reinforcement: enables security reinforcement for Alibaba Cloud Linux 2.x. For more information about CIS reinforcement, see CIS reinforcement. ■ CentOS 7.x **Note** CentOS 8.x and later are not supported.
Logon Type Key Pair	■ Use a key pair. You must select a key pair that is used to log on to the cluster when you create the cluster. If no key pair is available, click **create a key pair** to create one in the ECS console. For more information, see Create an SSH key pair. After the key pair is created, specify it as the credential to log on to the cluster. ■ Use a password. ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.

■ If you select **Add Existing Instance**, you must select ECS instances that are deployed in the specified region. Then, set the **Operating System**, **Logon Type**, and **Key Pair** parameters in the same way as you create ECS instances.

ii. Configure advanced settings for worker nodes.

Parameter	Description
Node Protection	Specify whether to enable node protection. **Note** By default, this check box is selected. Node protection prevents nodes from being accidentally deleted in the console or by calling the API. This prevents user errors.
User Data	For more information, see Overview of ECS instance user data.
Custom Image	You can select a custom image for your nodes. After you select the *faas_f3* image, all nodes in the cluster are deployed by using this image. For more information about how to create a custom image, see Create a Kubernetes cluster by using a custom image.  **Note** ■ Only custom images based on CentOS 7.x and Alibaba Cloud Linux 2.x are supported. ■ To use custom images, submit a ticket to apply to be added to the whitelist.
Custom Node Name	Specify whether to use a **custom node name**. A node name consists of a prefix, an IP substring, and a suffix. ■ Both the prefix and suffix can contain one or more parts that are separated by periods (.).These parts can contain lowercase letters, digits, and hyphens (-), and must start and end with a lowercase letter or digit. ■ The IP substring length specifies the number of digits to be truncated from the end of the returned node IP address. Valid values: 5 to 12. For example, if the node IP address is 192.1xx.x.xx, the prefix is aliyun.com, the IP substring length is 5, and the suffix is test, the node name will be aliyun.com00055test.
CPU Policy	Set the CPU policy. ■ none: This policy indicates that the default CPU affinity is used. This is the default policy. ■ static: This policy allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Taints	Add taints to the worker nodes in the ACK cluster.

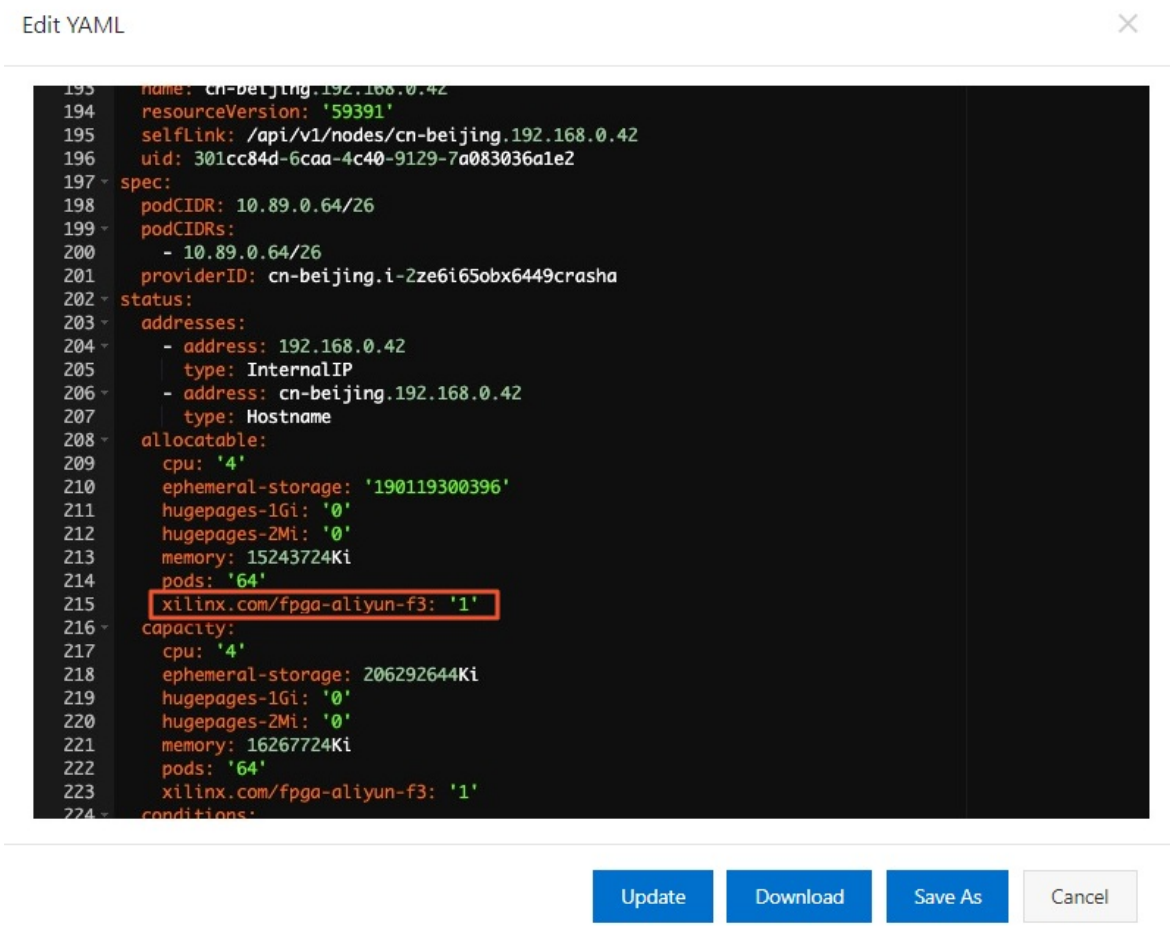
7. Click **Next: Component Configurations** to configure components.

Parameter	Description
Ingress	Specify whether to install Ingress controllers. By default, **Install Ingress Controllers** is selected. For more information, see Ingress高级用法. **Note** If you want to select **Create Ingress Dashboard**, you must first enable Log Service.
Volume Plug-in	Select a volume plug-in. FlexVolume and CSI are supported. ACK clusters can be automatically bound to Alibaba Cloud disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets that are mounted to pods. For more information, see Storage management-Flexvolume and Storage management-CSI.
Monitoring Agents	Specify whether to install the CloudMonitor agent. By default, **Install CloudMonitor Agent on ECS Instance** and **Enable Prometheus Monitoring** are selected. After the CloudMonitor agent is installed on ECS nodes, you can view monitoring data about the nodes in the CloudMonitor console.
Log Service	Specify whether to enable Log Service. You can select an existing Log Service project or create one. By default, **Enable Log Service** is selected. When you create an application, you can enable Log Service through a few steps. For more information, see Collect log files from containers by using Log Service. After you select **Enable Log Service**, you can specify whether to select **Create Ingress Dashboard** and **Install node-problem-detector and Create Event Center**.
Workflow Engine	Specify whether to enable Alibaba Cloud Genomics Compute Service (AGS). ○ If you select this check box, the system automatically installs the AGS workflow plug-in when the system creates the cluster. ○ If you clear this check box, you must manually install the AGS workflow plug-in. For more information, see Introduction to AGS CLI.

8. Click **Next: Confirm Order**.
9. Select **Terms of Service** and click **Create Cluster**.

Note It requires about 10 minutes for the system to create an ACK cluster that contains multiple nodes.

After the cluster is created, navigate to the **Clusters** page, find the created cluster, and then click the cluster name or click **Details** in the **Actions** column. In the left-side navigation pane, choose **Nodes > Nodes**. On the Nodes page, select a worker node and choose **More > Details** in the **Actions** column. On the page that appears, click **YAML** in the upper-right corner to view the FPGA resources that are mounted to the node.



16.2. GPU scheduling

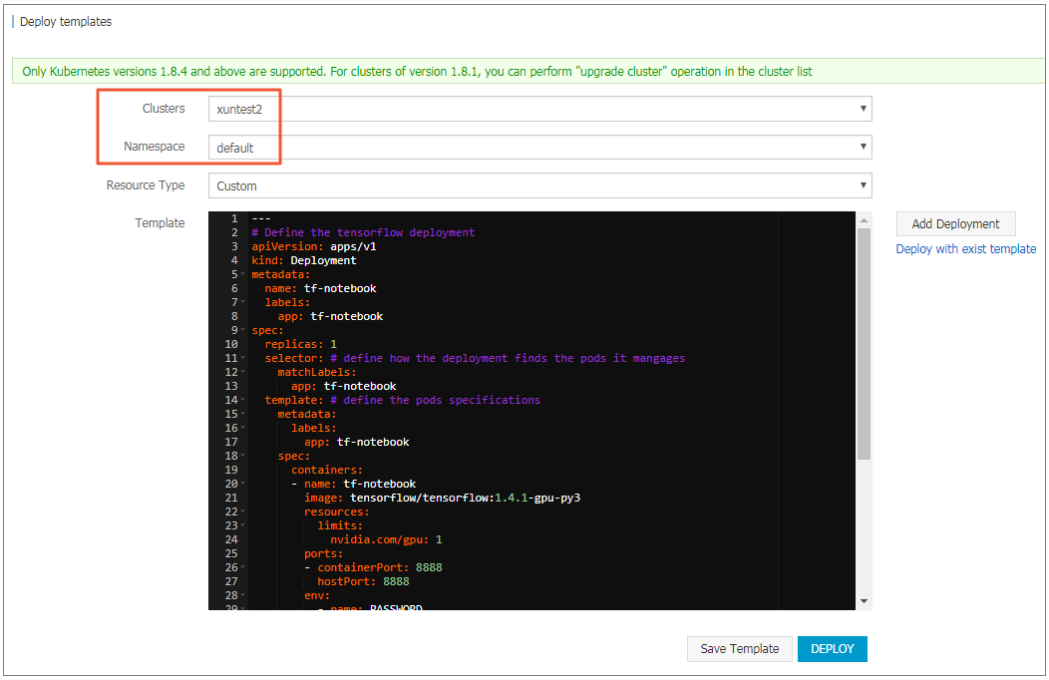
16.2.1. Use GPU scheduling for ACK clusters

This topic describes GPU scheduling for Container Service for Kubernetes (ACK) clusters with GPU-accelerated nodes by setting up a GPU-accelerated environment to run TensorFlow.

Set up a GPU-accelerated environment to run TensorFlow

In most cases, data scientists use Jupyter to set up an environment for TensorFlow. The following example shows how to deploy a Jupyter application.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Select the required cluster and namespace. Select a sample template, or set Sample Template to Custom and customize the template in the Template field. Then, click **Create** to create the application.



In this example, the template for the Jupyter application contains a Deployment and a Service.

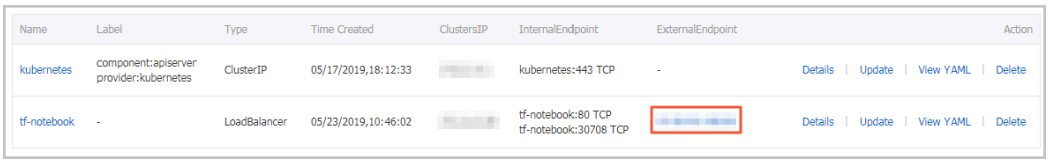
```
---
# Define the tensorflow deployment
apiVersion: apps/v1
kind: Deployment
metadata:
 name: tf-notebook
 labels:
  app: tf-notebook
spec:
 replicas: 1
 selector: # define how the deployment finds the pods it mangages
  matchLabels:
   app: tf-notebook
 template: # define the pods specifications
  metadata:
   labels:
    app: tf-notebook
  spec:
   containers:
   - name: tf-notebook
     image: tensorflow/tensorflow:1.4.1-gpu-py3
     resources:
      limits:
       nvidia.com/gpu: 1       #Specify the number of NVIDIA GPUs that are scheduled.
     ports:
     - containerPort: 8888
      hostPort: 8888
     env:
      - name: PASSWORD          #The password that is used to access the Jupyter application. You can modify
the password as needed.
       value: mypassword
# Define the tensorflow service
---
apiVersion: v1
kind: Service
metadata:
 name: tf-notebook
spec:
 ports:
 - port: 80
  targetPort: 8888
  name: jupyter
 selector:
  app: tf-notebook
 type: LoadBalancer                 #A Server Load Balancer (SLB) instance is used to route internal traffic and p
erform load balancing.
```

If you use a GPU scheduling solution provided by Kubernetes earlier than V1.9.3, you must define the volumes where the NVIDIA drivers are located.

```
volumes:
  - hostPath:
      path: /usr/lib/nvidia-375/bin
      name: bin
  - hostPath:
      path: /usr/lib/nvidia-375
      name: lib
```

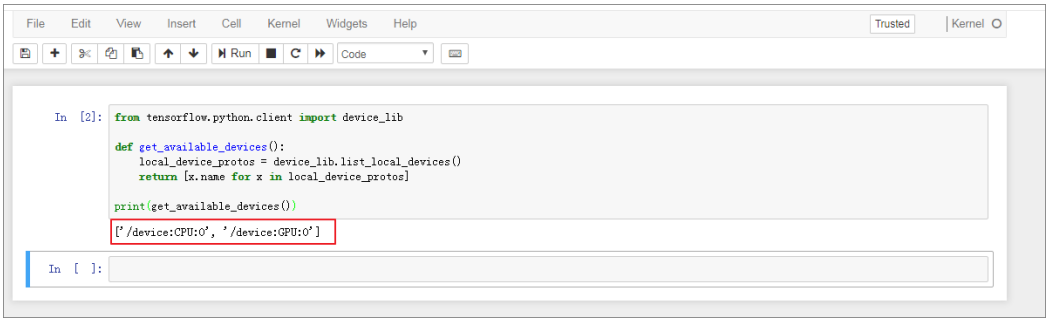
This solution requires you to manually modify the template if you want to schedule the GPUs to other clusters. However, in Kubernetes 1.9.3 and later, you do not need to set hostPath. The NVIDIA plug-in automatically identifies the links of the libraries and executable files that are required by NVIDIA drivers.

7. In the left-side navigation pane of the details page, choose **Network > Services**. Select the required cluster and namespace, find the tf-notebook Service, and then check its external endpoint.



8. To connect to the Jupyter application, enter `http://EXTERNAL-IP` into the address bar of your browser and enter the password specified in the template.
9. You can run the following program to verify that the Jupyter application is allowed to use GPU resources. The program lists all devices that can be used by TensorFlow:

```
from tensorflow.python.client import device_lib
def get_available_devices():
  local_device_protos = device_lib.list_local_devices()
  return [x.name for x in local_device_protos]
print(get_available_devices())
```



16.2.2. Use labels to schedule pods to GPU-accelerated nodes

To use Container Service for Kubernetes (ACK) clusters for GPU computing, you must schedule pods to GPU-accelerated nodes. ACK allows you to schedule pods to specific GPU-accelerated nodes by adding labels to the GPU-accelerated nodes.

Prerequisites

- An ACK cluster is created and GPU-accelerated nodes are added to the cluster. For more information, see Use GPU scheduling for ACK clusters.
- You are connected to a master node. You can check information such as node labels on the master node.

For more information, see Connect to Kubernetes clusters by using kubectl.

Context

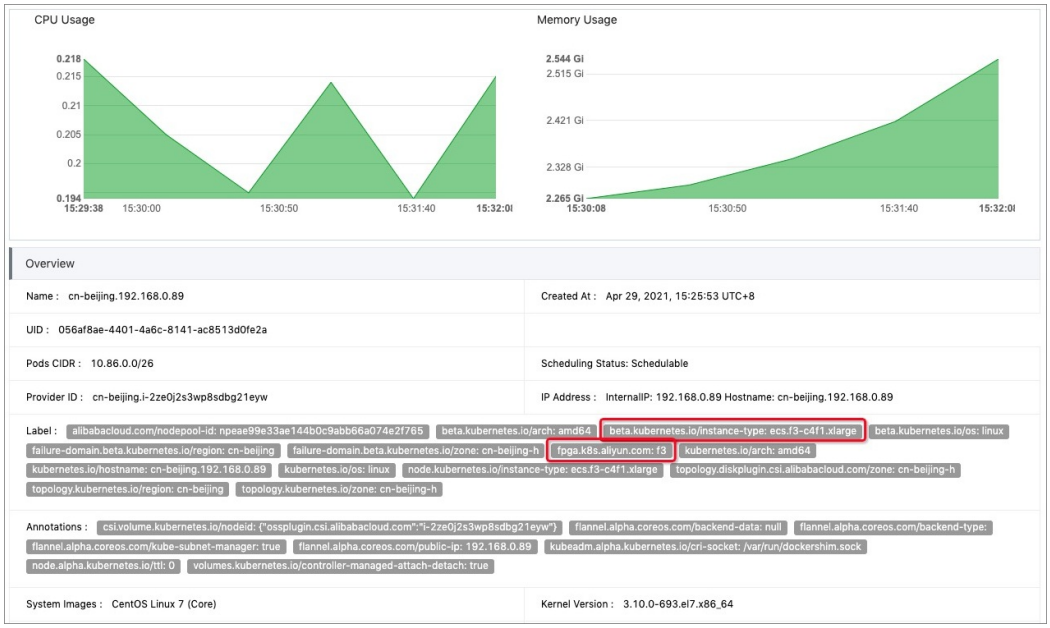
When ACK deploys nodes with NVIDIA GPUs, the attributes of these GPUs are discovered and exposed as node labels. These labels have the following benefits:

- You can use the labels to filter GPU-accelerated nodes.
- The labels can be used as conditions to schedule pods.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the **Nodes** page, find the GPU-accelerated node that you want to manage, and choose **More > Details** in the **Actions** column.

   Check the labels of the GPU-accelerated node.



You can also log on to a master node and run the following command to view the labels of GPU-accelerated nodes.
Run the following command:

```
kubectl get nodes
```

Response:

```
NAME                 STATUS   ROLES    AGE    VERSION
cn-beijing.i-2ze2dy2h9w97v65uuaft  Ready    master   2d     v1.11.2
cn-beijing.i-2ze8o1a45qdv5q8a7luz  Ready    <none>   2d     v1.11.2        #Compare these nodes with the no
des that are displayed in the ACK console to identify GPU-accelerated nodes.
cn-beijing.i-2ze8o1a45qdv5q8a7lv0  Ready    <none>   2d     v1.11.2
cn-beijing.i-2ze9xylyn11vop7g5bwe  Ready    master   2d     v1.11.2
cn-beijing.i-2zed5sw8snjniq6mf5e5  Ready    master   2d     v1.11.2
cn-beijing.i-2zej9s0zijykp9pwf7lu  Ready    <none>   2d     v1.11.2
```

Select a GPU-accelerated node and run the following command to query the labels of the node:

```
kubectl describe node cn-beijing.i-2ze8o1a45qdv5q8a7luz
```

Response:

```
Name:         cn-beijing.i-2ze8o1a45qdv5q8a7luz
Roles:        <none>
Labels:       aliyun.accelerator/nvidia_count=1                #Pay attention to this field.
              aliyun.accelerator/nvidia_mem=12209MiB
              aliyun.accelerator/nvidia_name=Tesla-M40
              beta.kubernetes.io/arch=amd64
              beta.kubernetes.io/instance-type=ecs.gn4-c4g1.xlarge
              beta.kubernetes.io/os=linux
              failure-domain.beta.kubernetes.io/region=cn-beijing
              failure-domain.beta.kubernetes.io/zone=cn-beijing-a
              kubernetes.io/hostname=cn-beijing.i-2ze8o1a45qdv5q8a7luz
 ......
```

In this example, the following labels are added to the GPU-accelerated node.

key	value
aliyun.accelerator/nvidia_count	The number of GPU cores.
aliyun.accelerator/nvidia_mem	The size of the GPU memory. Unit: MiB.
aliyun.accelerator/nvidia_name	The name of the NVIDIA GPU.

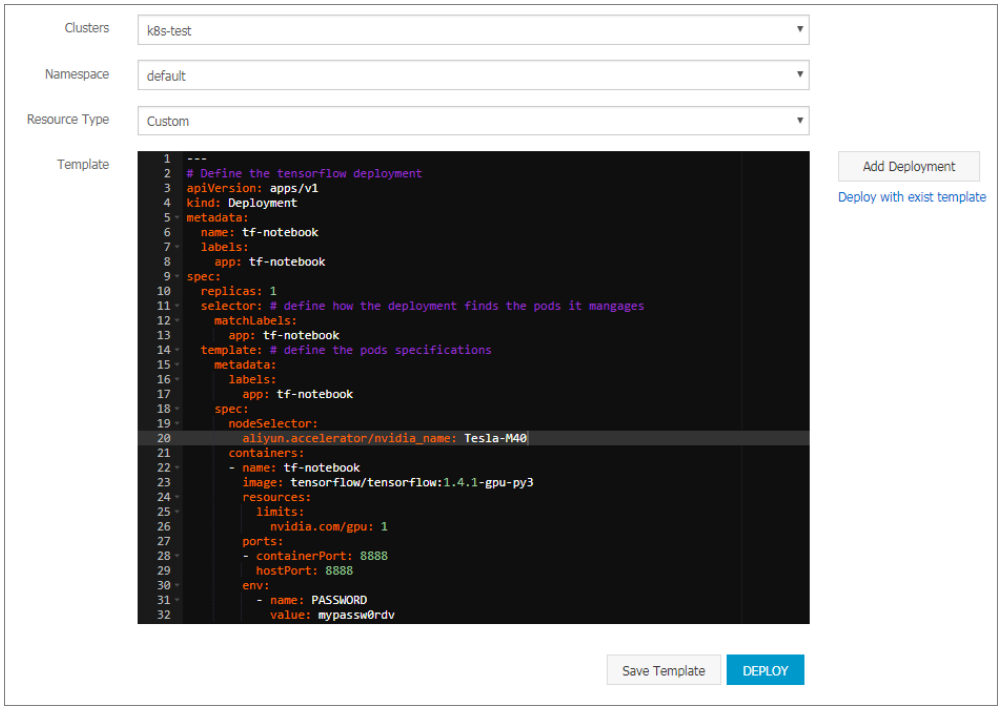
GPU-accelerated nodes of the same type have the same GPU name. You can use this label to locate GPU-accelerated nodes.
Run the following command:

```
kubectl get no -l aliyun.accelerator/nvidia_name=Tesla-M40
```

Response:

```
NAME                 STATUS   ROLES    AGE    VERSION
cn-beijing.i-2ze8o1a45qdv5q8a7luz  Ready    <none>   2d     v1.11.2
cn-beijing.i-2ze8o1a45qdv5q8a7lv0  Ready    <none>   2d     v1.11.2
```

6. In the left-side navigation pane of the details page, choose **Workloads > Deployments**. On the **Deployments** page, click **Create from YAML**.
   i. Create a Deployment for a TensorFlow job. The Deployment schedules pods to a GPU-accelerated node.




```yaml
---
# Define the tensorflow deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tf-notebook
  labels:
    app: tf-notebook
spec:
  replicas: 1
  selector: # define how the deployment finds the pods it mangages
    matchLabels:
      app: tf-notebook
  template: # define the pods specifications
    metadata:
      labels:
        app: tf-notebook
    spec:
      nodeSelector:
        aliyun.accelerator/nvidia_name: Tesla-M40
      containers:
      - name: tf-notebook
        image: tensorflow/tensorflow:1.4.1-gpu-py3
        resources:
          limits:
            nvidia.com/gpu: 1
        ports:
        - containerPort: 8888
          hostPort: 8888
        env:
          - name: PASSWORD
            value: mypassw0rdv
```

The following YAML template is used to create the Deployment:

```
---
# Define the tensorflow deployment
apiVersion: apps/v1
kind: Deployment
metadata:
 name: tf-notebook
 labels:
  app: tf-notebook
spec:
 replicas: 1
 selector: # define how the deployment finds the pods it mangages
  matchLabels:
   app: tf-notebook
 template: # define the pods specifications
  metadata:
   labels:
    app: tf-notebook
  spec:
   nodeSelector:                          #Pay attention.
    aliyun.accelerator/nvidia_name: Tesla-M40
   containers:
   - name: tf-notebook
    image: tensorflow/tensorflow:1.4.1-gpu-py3
    resources:
     limits:
      nvidia.com/gpu: 1                    #Pay attention.
    ports:
    - containerPort: 8888
     hostPort: 8888
    env:
     - name: PASSWORD
      value: mypassw0rdv
```

ii. You can also exclude an application from GPU-accelerated nodes. The following example shows how to schedule a pod based on node affinity for an NGINX application. For more information, see the section that describes node affinity in Create a stateless application by using a Deployment.

The following YAML template is used to schedule the pod:

```
apiVersion: v1
kind: Pod
metadata:
 name: not-in-gpu-node
spec:
 affinity:
  nodeAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
    - matchExpressions:
     - key: aliyun.accelerator/nvidia_name
      operator: DoesNotExist
 containers:
 - name: not-in-gpu-node
  image: nginx
```

7. In the left-side navigation pane of the details page, choose **Workloads > Pods**.
On the Pods page, you can find that the pods in the preceding examples are scheduled to the desired

nodes. This means that labels can be used to schedule pods to GPU-accelerated nodes.

16.2.3. Labels used by ACK to control GPUs

After ack-ai-installer is installed on a GPU-accelerated node, you can add specific labels to the node. This allows you to enable various features, such as GPU sharing, memory isolation, and topology-aware GPU scheduling. This topic describes the labels that you can use for GPU-accelerated nodes and the GPU-related features.

Labels for GPU-accelerated nodes

If you want to use cGPU or topology-aware GPU scheduling on a GPU-accelerated node, you must add specific labels to the node.

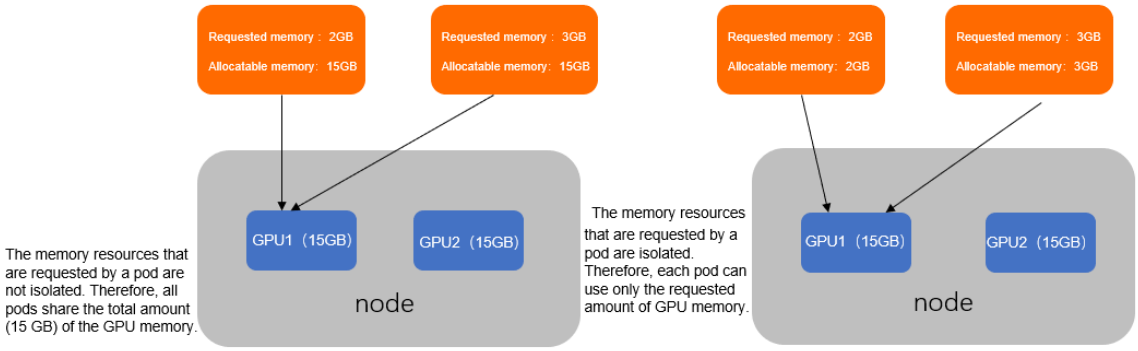
Notice If a label of a GPU-accelerated node is replaced by a mutually exclusive label, make sure that pods that request extended resources on the node have completed all operations. For example, before the label ack.node.gpu.schedule=share is changed to ack.node.gpu.schedule=topology on a GPU-accelerated node, you must make sure that all pods that request aliyun.com/gpu-mem resources on the node have completed all operations.

Label	Extended resource name	Description
ack.node.gpu.schedule=share	aliyun.com/gpu-mem	Enables only GPU sharing on a GPU-accelerated node. Memory isolation is disabled.
ack.node.gpu.schedule=cgpu	aliyun.com/gpu-mem	Enables both GPU sharing and memory isolation on a GPU-accelerated node.
ack.node.gpu.schedule=topology	aliyun.com/gpu	Enables topology-aware GPU scheduling on a GPU-accelerated node.
ack.node.gpu.schedule=default	nvidia.com/gpu	Enables the default GPU scheduling policy on a GPU-accelerated node.
ack.node.gpu.placement=binpack	\	Uses the binpack algorithm to schedule GPUs to pods. This applies only when cGPU is used.
ack.node.gpu.placement=spread	\	Uses the spread algorithm to schedule GPUs to pods. This applies only when cGPU is used.

GPU sharing and memory isolation

- The GPU sharing feature allows you to share one GPU of a node among multiple pods. For example, a node is installed with 2 GPUs (GPU 1 and GPU 2) and the total memory of each GPU is 15 GiB. When 2 pods (Pod 1 and Pod 2) use GPU 1 at the same time, GPU 1 is shared by Pod 1 and Pod 2.
- The memory isolation feature allows you to isolate the memory that is allocated from a GPU to each pod that shares the GPU.

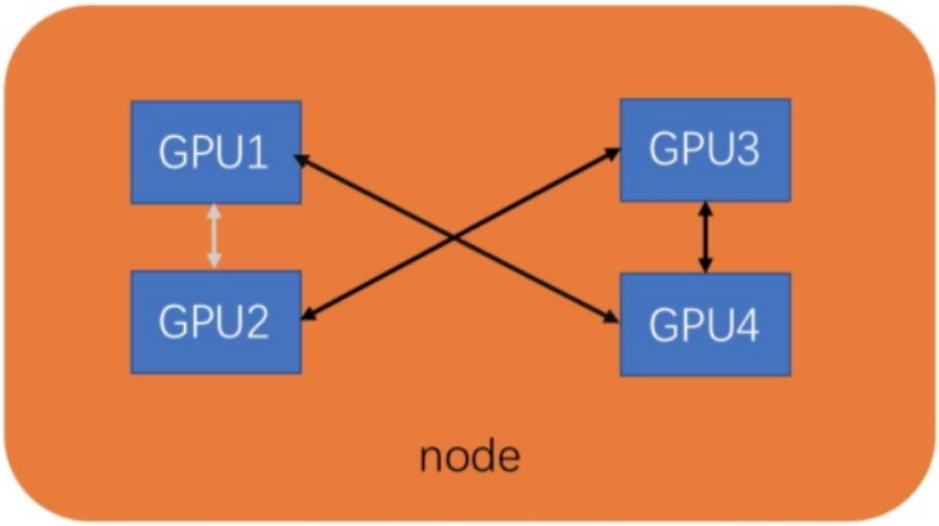
- Pod 1 requests 2 GiB of memory and Pod 2 requests 3 GiB of memory. When memory isolation is disabled, the amount of memory that can be used by Pod 1 or Pod 2 equals the total amount (15 GiB), as shown in the left part of the following figure. In this case, pods may fail to run. For example, if a pod that requests 2 GiB of memory from GPU 1 uses 15 GiB of memory when the pod is running, other pods that share GPU 1 fail to run.
- When memory isolation is enabled, each pod uses an exclusive and limited amount of GPU memory, as shown in the right part of the following figure. For example, if a pod requests 2 GiB of memory from GPU 1, the amount of memory that can be used by the pod cannot exceed 2 GiB when the pod is running. If the pod attempts to use 3 GiB of memory, the application stops running.



memory isolation not supported vs memory isolation supported

Topology-aware GPU scheduling

In the following figure, a node is installed with 4 GPUs (GPU 1, GPU2, GPU 3, and GPU 4). The bandwidth reaches the highest amount when GPU 1 communicates with GPU 2. The bandwidth reaches the lowest amount when GPU 1 communicates with GPU 4. If a pod requests 2 GPUs, you can enable topology-aware GPU scheduling to assign an optimal GPU combination to the pod. In this case, the bandwidth reaches the highest amount when GPU 1 communicates with GPU 2. This indicates that the latency is minimized when data is transmitted between GPU 1 and GPU 2. Therefore, GPU 1 and GPU 2 are allocated to the pod.

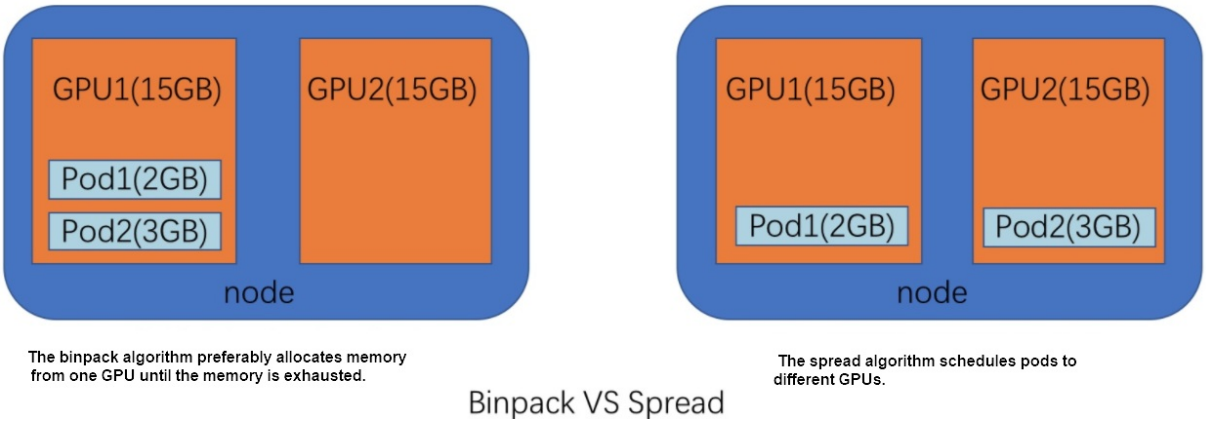


The binpack and spread algorithms

cGPU allows you to use the binpack and spread algorithms to allocate GPUs to pods.

In the following figure, a node is installed with 2 GPUs and the total memory of each GPU is 15 GiB, Pod 1 requests 2 GiB of GPU memory, and Pod 2 requests 3 GiB of GPU memory.

- If you use the binpack algorithm, the system preferably allocates memory from one GPU. Another GPU is used only after the memory of first GPU is exhausted. In this case, Pod 1 and Pod 2 are scheduled to GPU 1 in priority, as shown in the left part of the following figure.
- If you use the spread algorithm, the system attempts to schedule the pods to separate GPUs with best efforts. In this case, Pod 1 is scheduled to GPU 1 and Pod 2 is scheduled to GPU 2, as shown in the right part of the following figure.



16.2.4. Shared GPU scheduling

16.2.4.1. Overview

This topic describes Alibaba Cloud cGPU, which allows you to share GPU resources.

Background

Container Service for Kubernetes (ACK) provides the open source cGPU solution that allows you to run multiple containers on one GPU in an ACK cluster. You can enable cGPU for container clusters that are deployed on Alibaba Cloud, Amazon Web Services (AWS), Google Compute Engine (GCE), or data centers. cGPU reduces the expenses on GPUs. However, when you run multiple containers on one GPU, it is a challenge to ensure container stability.

To ensure the container stability, you must isolate the GPU resources that are assigned to each container. When you run multiple containers on one GPU, GPU resources are assigned to each container as required. However, if one container occupies excessive GPU resources, the performance of other containers may be affected. To solve this issue, many solutions have been provided in the computing industry. For example, NVIDIA virtual GPU (vGPU), Multi-Process Service (MPS), and vCUDA can enable fine-grained GPU resource allocation.

Features

The cGPU solution is provided by Alibaba Cloud. The cGPU solution uses the server kernel driver that is developed by Alibaba Cloud to provide more efficient use of the underlying drivers of NVIDIA GPUs. cGPU provides the following features:

- High compatibility: cGPU is compatible with standard open source solutions, such as Kubernetes and NVIDIA Docker.
- Ease-of-use: cGPU adopts a user-friendly design. To replace a Compute Unified Device Architecture (CUDA) library of an artificial intelligence (AI) application, you do not need to re-compile the application or create a new container image.
- Stability: cGPU provides stable underlying operations on NVIDIA GPUs. API operations on CUDA libraries and some private API operations on cuDNN are difficult to call.
- Resource isolation: cGPU ensures that the allocated GPU memory and computing capacity do not affect

each other.

Based on cGPU, ACK enables GPU sharing and the scheduling of multiple tasks to one GPU. This enables GPU sharing and memory isolation for scheduled Kubernetes resources and the container runtime. This provides low-cost, reliable, and user-friendly GPU sharing and memory isolation for large scale business.

Related information

- Install the cGPU component
- Enable GPU sharing
- Monitor and isolate GPU resources

16.2.4.2. Install the cGPU component

Container Service for Kubernetes (ACK) provides cGPU to enable GPU sharing and scheduling. You can use cGPU to share one GPU in model inference scenarios. In addition, the NVIDIA kernel driver ensures GPU memory isolation among containers. This topic describes how to install the resource isolation module and an inspection tool on a GPU-accelerated node. This enables GPU scheduling and memory isolation.

Prerequisites

- An ACK cluster with GPU-accelerated nodes is created.

  > **Note**
  > - Only dedicated Kubernetes clusters with GPU-accelerated nodes support the component described in this topic. Managed Kubernetes clusters with GPU-accelerated nodes do not support the component described in this topic.
  > - If you want to install the cGPU component in professional Kubernetes clusters, see Install and use ack-ai-installer and the GPU scheduling inspection tool.

- Connect to Kubernetes clusters by using kubectl

Limits

Item	Supported version
Kubernetes	1.12.6 and later. Only dedicated Kubernetes clusters are supported.
Helm	3.0 and later
NVIDIA driver	418.87.01 and later
Docker	19.03.5
Operating system	CentOS 7.x, Alibaba Cloud Linux 2.x, Ubuntu 16.04, and Ubuntu 18.04
GPU	Tesla P4, Tesla P100, Tesla T4, and Tesla v100

Step 1: Add labels to GPU-accelerated nodes

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or

click **Details** in the **Actions** column. The details page of the cluster appears.

4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner.
6. On the Labels tab of the **Manage Labels and Taints** page, select the nodes that you want to manage and click **Add Label**.
7. In the **Add** dialog box, set **Name** and **Value**.

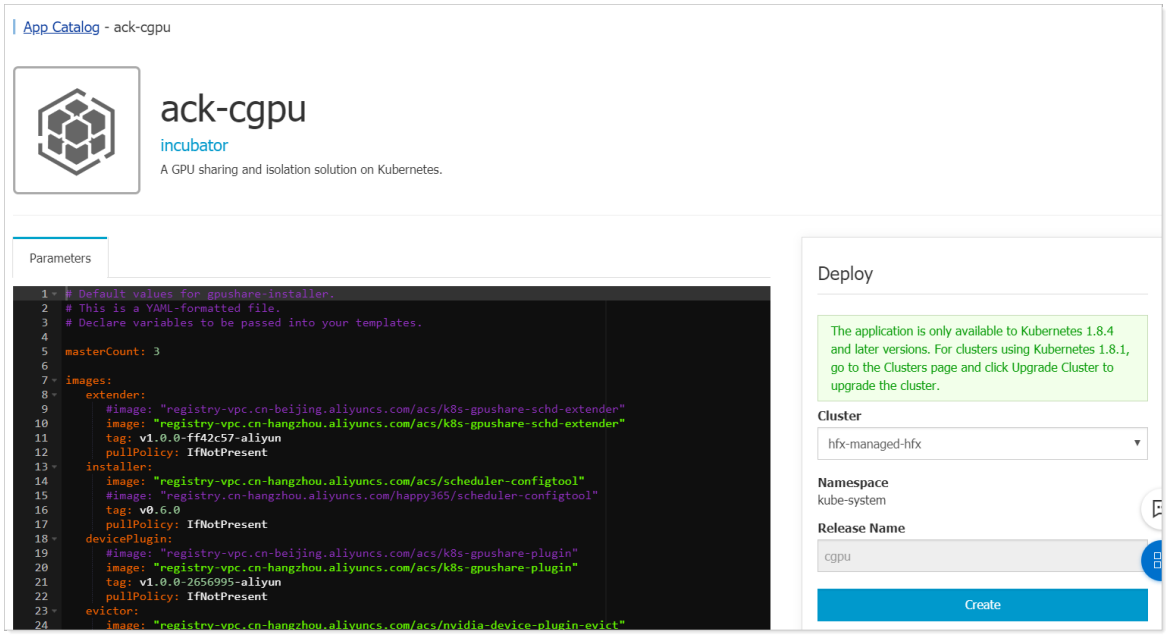
> **Notice**
> - You must set **Name** to **cgpu** and set **Value** to **true**.
> - If you delete the cgpu label, GPU sharing is still enabled. To disable GPU sharing, set **Name** to **cgpu** and set **Value** to **false**.

8. Click **OK**.

Step 2: Install the cGPU component on the labeled nodes

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, search for ack-cgpu and click ack-cgpu after it appears.
4. On the **App Catalog - ack-cgpu** page, select the cluster that you want to manage in the **Deploy** section and click **Create**.

   You do not need to set **Namespace** or **Release Name**. The default values are used.



You can run the `helm get manifest cgpu -n kube-system | kubectl get -f -` command to check whether the cGPU component is installed. If the following output is returned, it indicates that the cGPU component is installed.

```
helm get manifest cgpu -n kube-system | kubectl get -f -
```

```
NAME                    SECRETS  AGE
serviceaccount/gpushare-device-plugin  1      39s
serviceaccount/gpushare-schd-extender  1      39s
NAME                                   AGE
clusterrole.rbac.authorization.k8s.io/gpushare-device-plugin  39s
clusterrole.rbac.authorization.k8s.io/gpushare-schd-extender  39s
NAME                                    AGE
clusterrolebinding.rbac.authorization.k8s.io/gpushare-device-plugin  39s
clusterrolebinding.rbac.authorization.k8s.io/gpushare-schd-extender  39s
NAME                 TYPE     CLUSTER-IP   EXTERNAL-IP  PORT(S)       AGE
service/gpushare-schd-extender  NodePort  10.6.13.125  <none>     12345:32766/TCP  39s
NAME                      DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR   AGE
daemonset.apps/cgpu-installer        4     4     4    4       4       cgpu=true     39s
daemonset.apps/device-plugin-evict-ds   4     4     4    4       4       cgpu=true     39s
daemonset.apps/device-plugin-recover-ds  0     0     0    0       0       cgpu=false  39s
daemonset.apps/gpushare-device-plugin-ds  4     4     4    4       4       cgpu=true     39s
NAME                    READY  UP-TO-DATE  AVAILABLE  AGE
deployment.apps/gpushare-schd-extender  1/1   1       1      38s
NAME                COMPLETIONS  DURATION  AGE
job.batch/gpushare-installer  3/1 of 3    3s     38s
```

Related information

- Overview
- Enable GPU sharing
- Monitor and isolate GPU resources

16.2.4.3. Enable GPU sharing

This topic describes how to use cGPU to isolate GPU memory after you create containers that share one GPU by using YAML. This improves GPU resource usage.

Prerequisites

Install the cGPU component

Procedure

1. Run the following command to query information about GPU sharing in your cluster:

   ```
   kubectl inspect cgpu
   ```

   Expected output:

   ```
   NAME             IPADDRESS   GPU0(Allocated/Total)  GPU1(Allocated/Total)  GPU Memory(GiB)
   cn-shanghai.192.168.0.4  192.168.0.4  0/7            0/7            0/14
   ----------------------------------------------------------------------
   Allocated/Total GPU Memory In Cluster:
   0/14 (0%)
   ```

   > **Note** To query detailed information about GPU sharing, run the **kubectl inspect cgpu -d** command.

2. Use the following YAML file to create containers that share one GPU:

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
 name: binpack
 labels:
  app: binpack
spec:
 replicas: 1
 serviceName: "binpack-1"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: binpack-1
 template: # define the pods specifications
  metadata:
   labels:
    app: binpack-1
  spec:
   containers:
   - name: binpack-1
    image: registry.cn-shanghai.aliyuncs.com/tensorflow-samples/tensorflow-gpu-mem:10.0-runtime-centos7
     command:
      - python3
      - /app/main.py
     resources:
      limits:
       # GiB
       aliyun.com/gpu-mem: 3
```

Note aliyun.com/gpu-mem specifies the size of GPU memory.

3. Run the following command to query the result of resource scheduling performed by cGPU:

```
kubectl inspect cgpu
```

Expected output:

```
NAME                IPADDRESS     GPU0(Allocated/Total)  GPU Memory(GiB)
cn-beijing.192.168.1.105  192.168.1.105  3/14                   3/14
--------------------------------------------------------------------------
Allocated/Total GPU Memory In Cluster:
3/14 (21%)
```

Result

You can use one of the following methods to check whether cGPU has isolated the GPU memory that is allocated to different containers:

- Run the following command to view the log data of the application that is deployed in Step 2. You can check whether GPU memory is isolated by cGPU based on the log data.

  ```
  kubectl logs binpack-0 --tail=1
  ```

  Expected output:

```
2020-03-13 09:14:13.931003: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1326] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 2832 MB memory) -> physical GPU (device: 0, name: Tesla T4, pci bus id: 0000:00:07.0, compute capability: 7.5)
```

The output indicates that the container requests 2,832 MiB of GPU memory.

- Run the following command to log on to the container and view the amount of GPU memory that is allocated to the container:

```
kubectl exec -it binpack-0 nvidia-smi
```

Expected output:

```
Fri Mar 13 09:32:18 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:00:07.0 Off |                    0 |
| N/A  41C   P0    26W /  70W |   3043MiB /  3231MiB |     0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+ ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:00:07.0 Off |                    0 |
| N/A  41C   P0    26W /  70W |   3043MiB /  3231MiB |     0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

The output indicates that the amount of GPU memory allocated to the container is 3,231 MiB (3 x 1,024 = 3,072).

- Run the following command to view the total GPU memory of the host:

```
nvidia-smi
```

Expected output:

```
Fri Mar 13 17:36:24 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01   Driver Version: 418.87.01   CUDA Version: 10.1    |
|-------------------------------+----------------------+----------------------+
| GPU  Name      Persistence-M| Bus-Id      Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|  0  Tesla T4        On   | 00000000:00:07.0 Off |                    0 |
| N/A  40C   P0   26W /  70W |  3053MiB / 15079MiB |     0%    Default |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU     PID  Type  Process name                             Usage      |
|=============================================================================|
|   0    8796    C  python3                                    3043MiB |
+-----------------------------------------------------------------------------+
```

The output indicates that the total GPU memory of the host is 15,079 MiB and the amount of GPU memory that is allocated to the container is 3,053 MiB.

Related information

- Overview
- Monitor and isolate GPU resources

16.2.4.4. Monitor and isolate GPU resources

Container Service for Kubernetes (ACK) allows you to install the managed Prometheus plug-in. You can use the plug-in to monitor GPU resources. You can use the cGPU solution to schedule multiple applications to one GPU and isolate the GPU memory and computing power that are allocated to each application. This topic describes how to monitor GPU memory usage by using the managed Prometheus plug-in. This topic also describes how to isolate GPU resources by using cGPU.

Prerequisites

- A standard dedicated Kubernetes cluster with GPU-accelerated nodes is created and the Kubernetes version is 1.16 or later.
- Activate and upgrade ARMS.
- An Alibaba Cloud account is used to log on to the Resource Access Management (RAM) console. The account is authorized to use ARMS Prometheus.
- The GPU model is Tesla P4, Tesla P100, Tesla T4, or Tesla V100 (16 GB).

Context

The development of AI is fueled by high hash rates, large amounts of data, and optimized algorithms. NVIDIA GPUs provide common heterogeneous computing techniques. These techniques are the basis for high-performance deep learning. The cost of GPUs is high. If each application uses one dedicated GPU in prediction scenarios, computing resources may be wasted. GPU sharing improves resource usage. You must consider how to achieve the highest query rate at the lowest cost and how to fulfill the application service level agreement (SLA).

Use the managed Prometheus plug-in to monitor dedicated GPUs

1. Log on to the Application Real-Time Monitoring Service (ARMS) console.
2. In the left-side navigation pane, click **Prometheus Monitoring**.

3. On the **Prometheus Monitoring** page, select the region where the cluster is deployed and click **Install** in the **Actions** column.
4. In the **Confirmation** message, click **OK**.

   It requires about 2 minutes to install the Prometheus plug-in. After the Prometheus plug-in is installed, it appears in the **Installed Dashboards** column.
5. You can deploy the following sample application by using a CLI. For more information, see Manage applications by using commands.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: app-3g-v1
 labels:
  app: app-3g-v1
spec:
 replicas: 1
 serviceName: "app-3g-v1"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: app-3g-v1
 updateStrategy:
  type: RollingUpdate
 template: # define the pods specifications
  metadata:
   labels:
    app: app-3g-v1
  spec:
   containers:
   - name: app-3g-v1
    image: registry.cn-shanghai.aliyuncs.com/tensorflow-samples/cuda-malloc:3G
    resources:
     limits:
      nvidia.com/gpu: 1
```

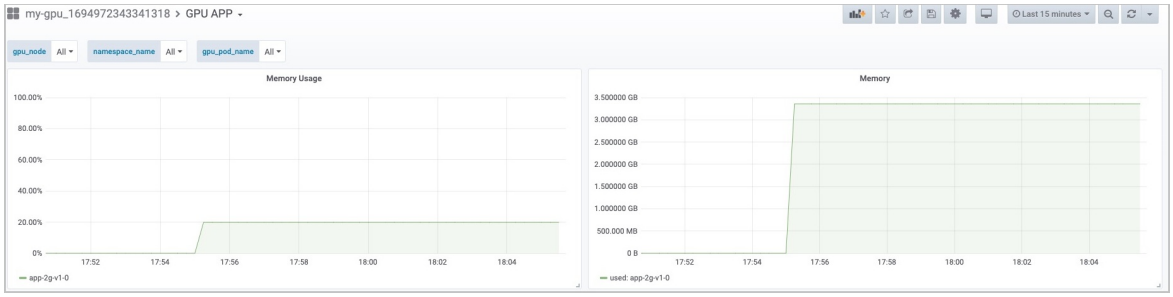
   After the application is deployed, run the following command to query the state of the application. The output indicates that the application name is app-3g-v1-0.

```
kubectl get po
```

   Expected output:

```
NAME     READY STATUS  RESTARTS AGE
app-3g-v1-0 1/1   Running 1     2m56s
```

6. Find and click the cluster where the application is deployed. On the Dashboards page, click **GPU APP** in the **Name** column.
   The following figure shows that the application uses only 20% of the GPU memory, which indicates that 80% of the GPU memory is wasted. The total GPU memory is about 16 GB. However, the memory usage stabilizes at about 3.4 GB. If you allocate one GPU to each application, a large amount of GPU resources are wasted. To improve GPU resource usage, you can use cGPU to share a GPU among multiple applications.



Share one GPU among multiple containers

1. Add labels to GPU-accelerated nodes.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster, or click **Applications** in the **Actions** column.
   iv. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
   v. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner of the page.
   vi. On the **Manage Labels and Taints** page, select the nodes to which you want to add a label and click **Add Label**.
   vii. In the **Add** dialog box, set Name to *cgpu*, set Value to *true*, and then click **OK**.

   > **Notice** If a worker node is added with the *cgpu=true* label, the GPU resource **nvidia.com/gpu** is not exclusive to the worker node. To disable GPU sharing for the worker node, set the value of cgpu to *false*. This makes the GPU resource **nvidia.com/gpu** exclusive to the worker node.

2. Install the cGPU component.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
   iii. On the **App Catalog** page, search for ack-cgpu and click ack-cgpu after it appears.
   iv. In the **Deploy** section on the right side of the page, select the cluster you created, select the namespace where you want to deploy ack-cgpu, and then click **Create**.
   v. Log on to a master node and run the following command to query GPU resources.

   For more information, see Connect to Kubernetes clusters by using kubectl.

   ```
   kubectl inspect cgpu
   ```

   Expected output:

   ```
   NAME                IPADDRESS     GPU0(Allocated/Total)  GPU Memory(GiB)
   cn-hangzhou.192.168.2.167  192.168.2.167  0/15                   0/15
   ------------------------------------------------------------------------------
   Allocated/Total GPU Memory In Cluster:
   0/15 (0%)
   ```

   > **Note** The output indicates that the GPU resources are switched from GPUs to GPU memory.

3. Deploy workloads that share GPU resources.
   i. Modify the *YAML* file that was used to deploy the sample application.
      - Modify the number of pod replicas from 1 to 2. This allows you to deploy two pods to run the application. Before you change the number of pod replicas, the GPU is exclusive to the pod. After you change the number of pod replicas, the GPU is shared by two pods.
      - Change the resource type from `nvidia.com/gpu` to `aliyun.com/gpu-mem`. The unit of GPU resources is changed to GB.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: app-3g-v1
 labels:
  app: app-3g-v1
spec:
 replicas: 2
 serviceName: "app-3g-v1"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: app-3g-v1
 template: # define the pods specifications
  metadata:
   labels:
    app: app-3g-v1
  spec:
   containers:
   - name: app-3g-v1
    image: registry.cn-shanghai.aliyuncs.com/tensorflow-samples/cuda-malloc:3G
    resources:
     limits:
      aliyun.com/gpu-mem: 4  # Each pod requests 4 GB of GPU memory. Two pod replicas are configure
d, therefore a total of 8 GB of GPU memory is requested by the application.
```

   ii. Recreate workloads based on the modified configurations.
   The output indicates that the two pods are scheduled to the same GPU-accelerated node.

```
kubectl inspect cgpu -d
```

   Expected output:

```
NAME:     cn-hangzhou.192.168.2.167
IPADDRESS: 192.168.2.167
NAME      NAMESPACE  GPU0(Allocated)
app-3g-v1-0 default   4
app-3g-v1-1 default   4
Allocated :  8 (53%)
Total :    15
---------------------------------------------------------
Allocated/Total GPU Memory In Cluster:  8/15 (53%)
```

iii. Run the following command to log on to two containers one after one.
The output indicates that the GPU memory limit is 4,301 MiB, which means that each container can use at most 4,301 MiB of GPU memory.

```
kubectl exec -it app-3g-v1-0 nvidia-smi
```

Expected output:

```
Mon Apr 13 01:33:10 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:07.0 Off |                    0 |
| N/A  37C    P0    57W / 300W |   3193MiB /  4301MiB |     0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

```
kubectl exec -it app-3g-v1-1 nvidia-smi
```

Expected output:

```
Mon Apr 13 01:36:07 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:07.0 Off |                    0 |
| N/A  38C    P0    57W / 300W |   3193MiB /  4301MiB |     0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

iv. Log on to the GPU-accelerated node to check the GPU usage.

The output indicates that the total used GPU memory is 6,396 MiB, which is the sum of the memory that is used by the two containers. This shows the cGPU solution has isolated GPU memory usage by container. If you log on to a container and apply for more GPU resources, a memory allocation error is returned.

a. Run the following command to log on to a GPU-accelerated node:

```
kubectl exec -it app-3g-v1-1 bash
```

b. Run the following command to query the GPU usage:

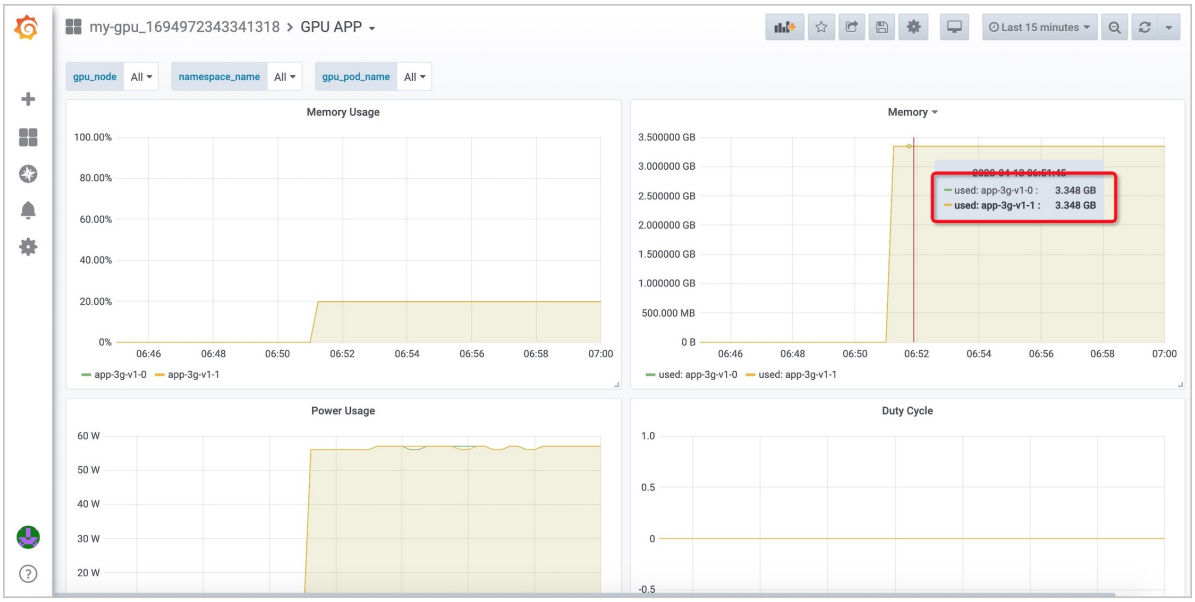
```
cuda_malloc -size=1024
```

Expected output:

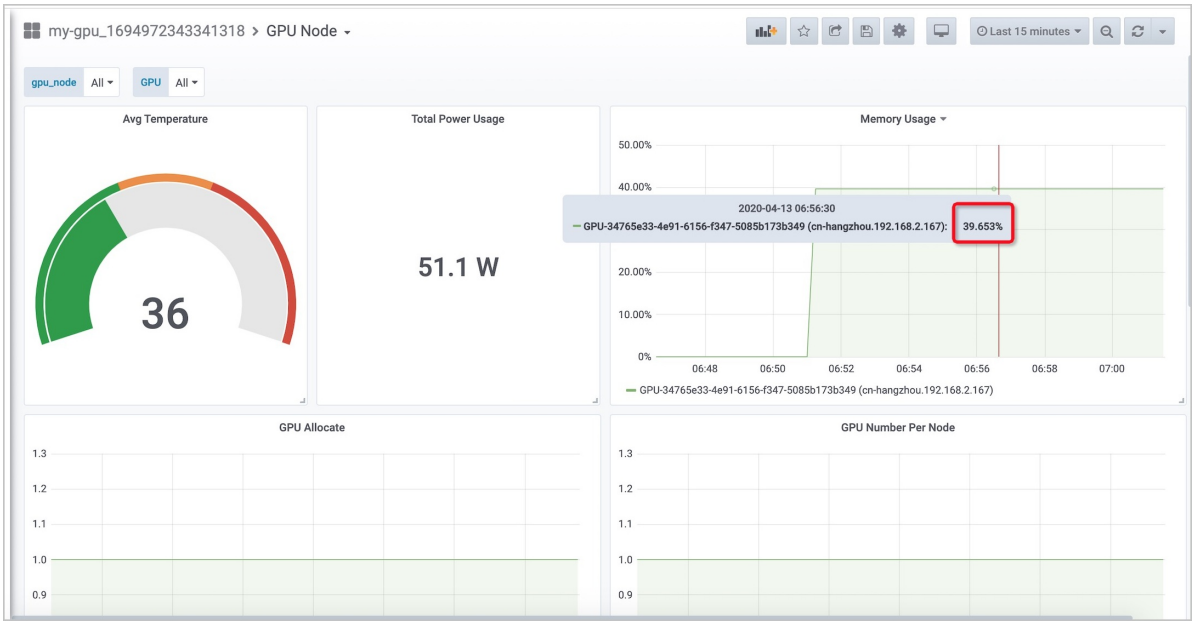
```
gpu_cuda_malloc starting...
Detected 1 CUDA Capable device(s)
Device 0: "Tesla V100-SXM2-16GB"
 CUDA Driver Version / Runtime Version       10.1 / 10.1
 Total amount of global memory:          4301 MBytes (4509925376 bytes)
Try to malloc 1024 MBytes memory on GPU 0
CUDA error at cgpu_cuda_malloc.cu:119 code=2(cudaErrorMemoryAllocation) "cudaMalloc( (void**
)&dev_c, malloc_size)"
```

You canmonitor the GPU usage of each application or node in the ARMS console.

- GPU APP: You can view the amount and percentage of GPU memory used by each application.



- GPU Node: You can view the memory usage of each GPU.



Use the managed Prometheus plug-in to monitor a shared GPU

If the amount of GPU memory requested by an application exceeds the upper limit, the GPU memory isolation module of the cGPU solution can prevent other applications from being affected.

1. Deploy a new application that uses the shared GPU.

   The application requests 4 GB of GPU memory. However, the actual memory usage of the application is 6 GB.

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
 name: app-6g-v1
 labels:
  app: app-6g-v1
spec:
 replicas: 1
 serviceName: "app-6g-v1"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: app-6g-v1
 template: # define the pods specifications
  metadata:
   labels:
    app: app-6g-v1
  spec:
   containers:
   - name: app-6g-v1
    image: registry.cn-shanghai.aliyuncs.com/tensorflow-samples/cuda-malloc:6G
    resources:
     limits:
      aliyun.com/gpu-mem: 4 # Each pod requests 4 GB of GPU memory. One pod replica is configured for the application, therefore a total of 4 GB of GPU memory is requested by the application.
```

2. Run the following command to query the state of the pod.

   The pod that runs the new application remains in the **CrashLoopBackOff** state. The two existing pods

are running as normal.

```
kubectl get pod
```

Expected output:

```
NAME      READY  STATUS       RESTARTS  AGE
app-3g-v1-0  1/1   Running      0      7h35m
app-3g-v1-1  1/1   Running      0      7h35m
app-6g-v1-0  0/1   CrashLoopBackOff  5      3m15s
```

3. Run the following command to check errors in the container log.

   The output indicates that the cudaErrorMemoryAllocation error has occurred.

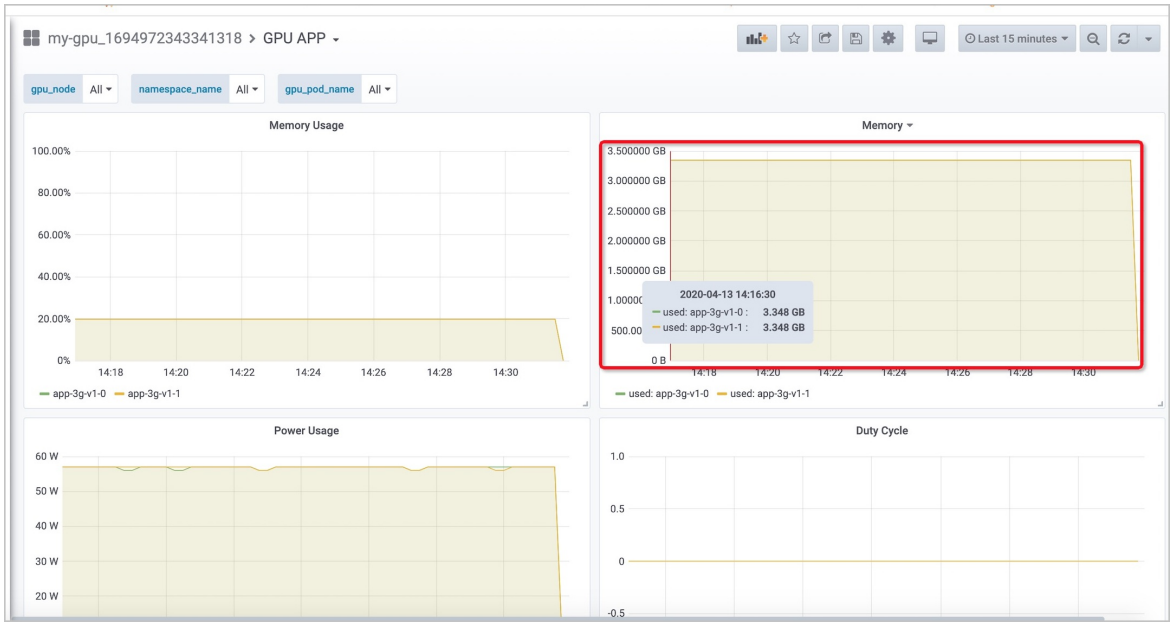
   ```
   kubectl logs app-6g-v1-0
   ```

   Expected output:

   ```
   CUDA error at cgpu_cuda_malloc.cu:119 code=2(cudaErrorMemoryAllocation) "cudaMalloc( (void**)&dev_c, malloc_size)"
   ```

4. Use the GPU APP dashboard provided by the managed Prometheus plug-in to view the states of containers.

   The following figure shows that existing containers are not affected after the new application is deployed.



16.2.4.5. Disable the memory isolation capability of cGPU

This topic describes how to disable the memory isolation capability of cGPU by using a sample application.

Prerequisites

Install the cGPU component

Procedure

1. Run the following command to query information about GPU sharing in your cluster:

   ```
   kubectl inspect cgpu
   ```

```
NAME              IPADDRESS     GPU0(Allocated/Total)  GPU Memory(GiB)
cn-beijing.192.16x.x.xx3  192.16x.x.xx3  0/15           0/15
cn-beijing.192.16x.x.xx1  192.16x.x.xx1  0/15           0/15
cn-beijing.192.16x.x.xx2  192.16x.x.xx2  0/15           0/15
------------------------------------------------------------------------------
Allocated/Total GPU Memory In Cluster:
0/45 (0%)
```

> **Note** To query detailed information about GPU sharing, run the `kubectl inspect cgpu -d` command.

2. Use the following YAML template to create a container for which GPU sharing is enabled and memory isolation is disabled:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: binpack
 labels:
  app: binpack
spec:
 replicas: 1
 serviceName: "binpack-1"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: binpack-1
 template: # define the pods specifications
  metadata:
   labels:
    app: binpack-1
  spec:
   containers:
   - name: binpack-1
    image: registry.cn-shanghai.aliyuncs.com/tensorflow-samples/tensorflow-gpu-mem:10.0-runtime-centos7
    command:
     - python3
     - /app/main.py
    env:
     - name: CGPU_DISABLE #Disable the memory isolation capability of cGPU.
      value: "true"
    resources:
     limits:
      # GiB
      aliyun.com/gpu-mem: 3
```

> **Note**
>
> - aliyun.com/gpu-mem: specifies the size of GPU memory.
> - To disable the memory isolation capability of cGPU, set CGPU_DISABLE to true.

3. Run the following command to query the result of GPU scheduling performed by cGPU:

```
kubectl inspect cgpu
```

```
NAME                IPADDRESS     GPU0(Allocated/Total)  GPU Memory(GiB)
cn-beijing.192.16x.x.xx1  192.16x.x.xx1  0/15               0/15
cn-beijing.192.16x.x.xx2  192.16x.x.xx2  0/15               0/15
cn-beijing.192.16x.x.xx3  192.16x.x.xx3  3/15               3/15
------------------------------------------------------------------------
Allocated/Total GPU Memory In Cluster:
3/45 (6%)
```

The newly created container is allocated with 3 GiB of GPU memory from the cn-beijing.192.16x.x.xx3 node.

Check the result

You can use one of the following methods to check whether the memory isolation capability of cGPU is disabled:

- Method 1: Run the following command to query the application log:

```
kubectl logs binpack-0 --tail=1
```

```
2020-08-25 08:14:54.927965: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1326] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 15024 MB memory) -> physical GPU (device: 0, name: Tesla V100-SXM2-16GB, pci bus id: 0000:00:07.0, compute capability: 7.0)
```

The returned log entry shows that the GPU memory available for the containerized application is 15,024 MiB. This indicates that the memory isolation capability of cGPU is disabled. If the memory isolation capability of cGPU is enabled, the amount of GPU memory that the containerized application can use is 3 GiB.

- Method 2: Run the following command to log on to the container and view the amount of GPU memory that is allocated to the container:

```
kubectl exec binpack-0 nvidia-smi
```

```
Tue Aug 25 08:23:33 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:00:07.0 Off |                    0 |
| N/A   33C    P0    55W / 300W |  15453MiB / 16130MiB |      1%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

The output shows that the GPU memory capacity of the host is 16,130 MiB and the amount of GPU memory that is allocated to the container is 15,453 MiB. This indicates that the memory isolation capability of cGPU is disabled. If the memory isolation capability of cGPU is enabled, the amount of GPU memory that is allocated to the container is 3 GiB.

Related information

- Overview
- Install the cGPU component
- Monitor and isolate GPU resources

16.2.5. Topology-aware GPU scheduling

Kubernetes does not support topology-aware GPU scheduling. In this case, GPUs are selected at random. The performance of GPU acceleration for training jobs can vary based on different combinations of GPUs. To resolve this issue, Container Service for Kubernetes (ACK) supports topology-aware GPU scheduling based on the scheduling framework. This feature selects a combination of GPUs from GPU-accelerated nodes to achieve optimal GPU acceleration for training jobs.
For more information about how to use topology-aware GPU scheduling, see the following topics:

- Overview
- Install the ack-ai-installer component
-
-

16.3. Observability

16.3.1. Monitor GPU errors

This topic describes how to use Kubernetes Event Center to monitor GPU-accelerated instances and configure alerts for Xid messages that indicate GPU errors. This provides diagnostic information that can be used for debugging reported NVIDIA driver errors.

Prerequisites

- A managed or dedicated GPU cluster is created. For more information, see Create a managed GPU cluster or Create a dedicated Kubernetes cluster with GPU-accelerated nodes.
- Create and use a Kubernetes event center

Context

An Xid message is an error report from the NVIDIA driver. Such a report is printed to the kernel log or event log of the operating system. An Xid message indicates that a general GPU error occurred. In most cases, a GPU error occurs due to improper driver programming over the GPU or due to the corruption of the commands sent to the GPU. You can use Xid messages to identify hardware, NVIDIA software, or application issues.

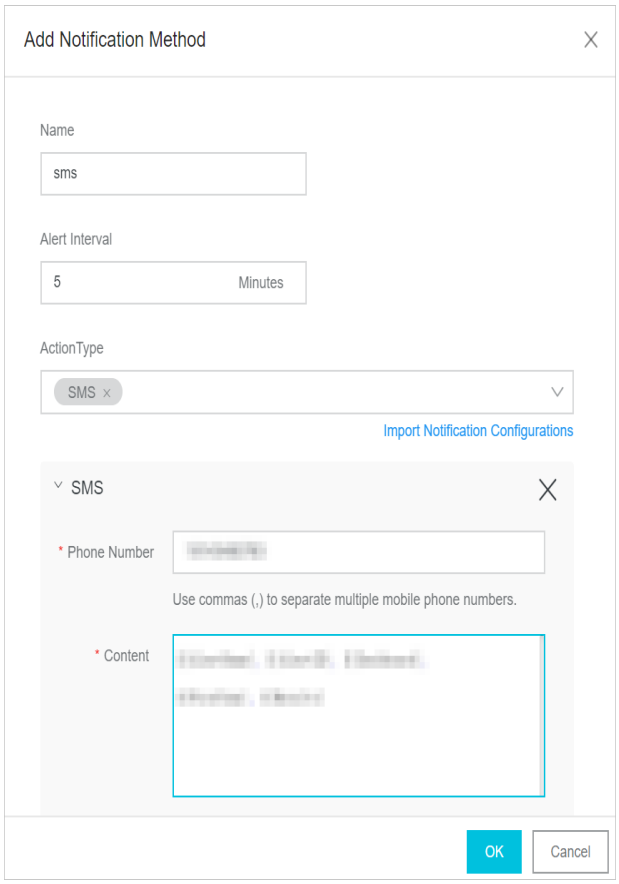
GPU drivers are prone to Xid errors. You can use Kubernetes Event Center to monitor Xid errors and configure alerts. This allows you to identify and troubleshoot issues at the earliest opportunity.

Procedure

1. Log on to the Log Service console. In the Log Application section, click the **K8s Event Center** card.

   For more information, see Create and use a Kubernetes event center.
2. In the left-side navigation pane under **K8s Event Center**, select the cluster that you want to manage, and click **Event Overview**.
   On the **Event Overview** tab, you can view Xid messages and triggered alerts.
3. In the left-side navigation pane, select the cluster that you want to manage, and click **Alert Configuration**.
4. Click **Add Notification Method**. In the **Add Notification Method** panel, configure the notification method, and click **OK**.

You can select a notification method to receive alerts through text messages, emails, or DingTalk notifications. You can also customize the alert content. The following figure shows how to enable SMS alerting.



5. After you configure the notification method, click **Modify** in the upper-right corner of the **Events** section. In the **Kubernetes GPU Xid Alerts** card, turn on the switch, and select **SMS** from the drop-down list.
6. In the **Events** section, click **Save**.
   After an alert is triggered, you receive a text message from Alibaba Cloud.

16.4. Operations & Maintenance (O&M) Management

16.4.1. Upgrade the Docker runtime of a GPU node

To isolate GPU resources shared by multiple nodes in a Kubernetes cluster, Docker 19.03.5 and its nvidia-container-runtime binary must be used. If the Docker runtime version is earlier than 19.03.5, you must upgrade the Docker runtime to Docker 19.03.5. This topic describes how to upgrade the Docker runtime and its nvidia-container-runtime binary on these nodes to ensure GPU sharing among these nodes.

Context

The nvidia-container-runtime binary allows you to build and run GPU-accelerated Docker containers. The binary automatically configures the containers. This ensures that the containers can use NVIDIA GPU resources.

Procedure

> **Notice** The steps described in this topic apply to only CentOS and Alibaba Cloud Linux 2.

Before you perform the following steps, you must use the command-line interface (CLI) to connect to your Container Service for Kubernetes (ACK) cluster. For more information, see Connect to Kubernetes clusters by using kubectl.

1. Run the following command on the master node to disconnect a specified node from the cluster.

   You must label the node as unschedulable. This prevents pods from being scheduled to the node during the upgrade.

   ```
   kubectl cordon <NODE_NAME>
   ```

   > **Note** <NODE_NAME> specifies the name of the node on which the Docker runtime will be upgraded. You can run the **kubectl get nodes** command to query node names.

2. Run the following command on the master node to migrate the pods on the node.

   After the node is disconnected from the cluster, you must migrate the pods on the node to other available nodes.

   ```
   kubectl drain <NODE_NAME> --ignore-daemonsets --delete-local-data --force
   ```

   > **Note** <NODE_NAME> specifies the name of the node on which the Docker runtime will be upgraded.

3. Run the following commands to stop kubelet and the Docker runtime.

   Stop kubelet and the Docker runtime before you upgrade the Docker runtime on the node.

   ```
   service kubelet stop
   docker rm -f $(docker ps -aq)
   service docker stop
   ```

4. Run the following commands to remove the Docker runtime and nvidia-container-runtime.

   Before you upgrade the Docker runtime and nvidia-container-runtime on the node, the current versions must be removed.

   ```
   yum remove -y docker-ce docker-ce-cli containerd
   yum remove -y nvidia-container-runtime*  libnvidia-container*
   ```

5. Run the following commands to back up and remove the *daemon.json* file:

```
cat /etc/docker/daemon.json
{
  "default-runtime": "nvidia",
  "runtimes": {
    "nvidia": {
      "path": "/usr/bin/nvidia-container-runtime",
      "runtimeArgs": []
    }
  },
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "10"
  },
  "bip": "169.254.123.1/24",
  "oom-score-adjust": -1000,
  "storage-driver": "overlay2",
  "storage-opts":["overlay2.override_kernel_check=true"],
  "live-restore": true
}
mv /etc/docker/daemon.json /tmp
```

6. Run the following command to install the Docker runtime.

   Download the Docker installation package to the node on which you want to upgrade the Docker runtime.

   ```
   VERSION=19.03.5
   URL=http://aliacs-k8s-cn-beijing.oss-cn-beijing.aliyuncs.com/public/pkg/docker/docker-${VERSION}.tar.gz
   curl -ssL $URL -o /tmp/docker-${VERSION}.tar.gz
   cd /tmp
   tar -xf docker-${VERSION}.tar.gz
   cd /tmp/pkg/docker/${VERSION}/rpm
   yum localinstall -y $(ls .)
   ```

7. Run the following command to install nvidia-container-runtime on the node:

   ```
   cd /tmp
   yum install -y unzip
   wget http://kubeflow.oss-cn-beijing.aliyuncs.com/nvidia.zip
   unzip nvidia.zip
   yum -y -q --nogpgcheck localinstall /tmp/nvidia/*
   ```

8. Run the following command to configure the *daemon.json* file.

   Overwrite the */etc/docker/daemon.json* file with the specified *daemon.json* file to make the original configurations take effect.

```
mv /tmp/daemon.json  /etc/docker/daemon.json
cat /etc/docker/daemon.json
{
  "default-runtime": "nvidia",
  "runtimes": {
    "nvidia": {
      "path": "/usr/bin/nvidia-container-runtime",
      "runtimeArgs": []
    }
  },
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "10"
  },
  "bip": "169.254.123.1/24",
  "oom-score-adjust": -1000,
  "storage-driver": "overlay2",
  "storage-opts":["overlay2.override_kernel_check=true"],
  "live-restore": true
}
```

9. Run the following commands to restart the Docker runtime and kubelet:

```
service docker start
service kubelet start
```

10. Run the following command to connect the node to the cluster.

    After the Docker runtime of the node is upgraded, the node is changed to the schedulable state in the cluster.

```
kubectl uncordon <NODE_NAME>
```

> **Note** <NODE_NAME> specifies the name of the node on which the Docker runtime has been upgraded.

11. Run the following command to restart the GPU installer on the node:

```
docker ps |grep cgpu-installer | awk '{print $1}' | xargs docker rm -f
```

Related information

- Overview
- Install the cGPU component

16.4.2. Upgrade the cGPU version on a node

Container Service for Kubernetes (ACK) supports GPU sharing. To enable GPU sharing, you must install cGPU on a node. This topic describes how to upgrade the cGPU version on a node by using a CLI and the ACK console.

Prerequisites

- Your machine is connected to the cluster by using kubectl. For more information, see Use kubectl to connect to an ACK cluster.
- ack-cgpu is installed in the cluster. For more information, see Install ack-cgpu.

- No workload is running on the node that you want to upgrade.

Upgrade the cGPU version on a node by using a CLI

The cgpu-installer component runs as a DaemonSet, which is used to install cGPU on nodes. To upgrade cGPU, you must change the image version of cgpu-installer to the version to which you want to upgrade.

The following cGPU image versions are supported:

- v0.8.10
- v0.8.12
- v0.8.13

> **Note** During the upgrade process, the node where cGPU is deployed is restarted. Make sure that no workload is running on the node before you upgrade cGPU.

1. Run the following command to modify the image version of cgpu-installer.

   ```
   kubectl edit ds cgpu-installer -n kube-system
   ```

   In this example, the image version is changed to V0.8.10.

   ```
   spec:
     containers:
     - env:
       - name: CHECK_REGIONS
         value: "true"
       - name: SUPPORT_REGIONS
         value: cn-hangzhou,cn-beijing,cn-shanghai,cn-zhangjiakou,cn-shenzhen,cn-chengdu,us-east-1,cn-heyuan,ap-southeast-1,ap-south

       image: registry-vpc.cn-beijing.aliyuncs.com/acs/cgpu-installer:v0.8.10
       imagePullPolicy: IfNotPresent
       name: cgpu-installer
       resources: {}
       securityContext:
   ```

2. Uninstall the earlier version of cGPU.
   i. Log on to the node. For more information about how to log on to a node, see Connect to a Linux instance by using password authentication.
   ii. Run the following command to stop Docker:

   ```
   systemctl stop docker
   ```

   iii. Run the following command to uninstall cGPU:

   ```
   bash /usr/local/cgpu-installer/uninstall.sh
   ```

   > **Note**
   > If */usr/local/cgpu-installer/uninstall.sh* does not exist, run the following command to uninstall the earlier version of cGPU.
   >
   > ```
   > wget http://aliacs-k8s-cn-beijing.oss-cn-beijing.aliyuncs.com/gpushare/cgpu-uninstall.sh -O /usr/local/cgpu-installer/uninstall.sh
   > ```

3. Restart the node. For more information about how to restart a node, see Restart an instance.

Verify the result

After the node is restarted, log on to the node and run the following command to query the cGPU version:

```
cat /proc/cgpu_km/version
```

Expected output:

```
0.8.10
```

The output indicates that the cGPU version is upgraded to V0.8.10.

Upgrade the cGPU version on a node by using the ACK console

The cgpu-installer component runs as a DaemonSet, which is used to install cGPU on nodes. To upgrade cGPU, you must change the image version of cgpu-installer to the version to which you want to upgrade.

The following cGPU image versions are supported:

- v0.8.10
- v0.8.12
- v0.8.13

1. Run the following command to modify the image version of cgpu-installer.

   ```
   kubectl edit ds cgpu-installer -n kube-system
   ```

   In this example, the image version is changed to V0.8.10.

2. Remove the node whose cGPU is to be upgraded from the cluster.

   For more information, see Remove nodes from an ACK cluster.

   i. Log on to the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
   iv. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
   v. On the **Nodes** page, select the node that you want to remove and click **Batch Remove**.
   vi. In the **Remove Node** dialog box, select **Drain the Node**.
   vii. Click **OK**.

3. Create a node pool.

   Create a node pool and add the node that you removed to the node pool.
   For more information, see 管理节点池.

   i. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
   ii. On the **Node Pools** page, click **Create Node Pool**.

iii. In the **Create Node Pool** dialog box, configure the parameters.

For more information about the parameters, see Create a dedicated Kubernetes cluster. The following list describes some of the parameters:

Parameter	Description	Recommended value
Quantity	Specify the initial number of nodes in the node pool.	In this example, set Quantity to 0.
Node Label	You can add labels to nodes in the node pool.	■ If ack-ai-installer is installed in the cluster, set **Key** to *ack.node.gpu.schedule* and **Value** to *cgpu*. ■ If ack-cgpu is installed in the cluster, set **Key** to *cgpu* and **Value** to *true*.

iv. Click **Confirm Order** to create the node pool.

4. Add the node to the node pool.

After the node pool is created, you must add the node that you removed in Step 2 to the node pool that you created in Step 3. For more information, see Add existing ECS instances to an ACK cluster.

Verify the result

After the node is added to the node pool, verify that the cGPU version is upgraded.

1. Run the following command to query the pods that run cgpu-installer on the node:

```
kubectl get po -l name=cgpu-installer -n kube-system -o wide
```

Expected output:

```
NAME          READY  STATUS   RESTARTS  AGE   IP        NODE          NOMINATED NODE  READINESS GA
TES
cgpu-installer-kkmp6  1/1   Running  0     4d2h  192.168.XXX.XX1  cn-beijing.192.168.XXX.XX1 <none>     <
none>
cgpu-installer-**2   1/1   Running  0     4d2h  192.168.XXX.XX2  cn-beijing.192.168.XXX.XX2 <none>     <no
ne>
cgpu-installer-**3   1/1   Running  0     4d2h  192.168.XXX.XX3  cn-beijing.192.168.XXX.XX3 <none>     <no
ne>
```

2. Run the following command to query the pod named `cgpu-installer-kkmp6`:

```
kubectl exec -ti cgpu-installer-kkmp6 -n kube-system -- bash
```

3. Run the following command to query the current cGPU version:

```
nsenter -t 1 -i -p -n -u -m -- cat /proc/cgpu_km/version
```

Expected output:

```
0.8.10
```

The output indicates that the cGPU version is V0.8.10.

16.4.3. Troubleshoot issues in GPU monitoring

When exceptions occur or no records are found on GPU monitoring dashboards, you can perform the steps in this topic to troubleshoot the issues.

Procedure

Step 1: Check whether GPU nodes exist in the cluster

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the **Nodes** page, check whether GPU nodes exist in the cluster.

> **Note** On the **Nodes** page, if the values in the **Configuration** column contains ****ecs.gn****, the cluster has GPU nodes.

Step 2: Check whether ack-arms-prometheus is installed

1. Check whether ack-arms-prometheus is installed in the cluster. For more information, see Enable ARMS Prometheus.
2. If ack-arms-prometheus is installed, run the following command to query the pods of ack-arms-prometheus:

```
kubectl get pods -n arms-prom
```

Expected output:

```
NAME                                        READY  STATUS   RESTARTS  AGE
arms-prom-ack-arms-prometheus-866cfd9f8f-x8jxl  1/1   Running  0     26d
```

If the pods are in the Running state, the pods run as expected. If the pods are not in the Running state, run the `kubectl describe pod` command to query the reason why the pods are not running.

Step 3: Check whether ack-prometheus-gpu-exporter is deployed

Run the following command to query the status and quantity of pods:

```
kubectl get pods -n arms-prom
```

Expected output:

```
NAME                                READY  STATUS   RESTARTS  AGE
ack-prometheus-gpu-exporter-6kpj7    1/1   Running  0     7d19h
ack-prometheus-gpu-exporter-bkbf8    1/1   Running  0     18h
ack-prometheus-gpu-exporter-blbnq    1/1   Running  0     18h
```

The preceding output indicates that the number of pods is the same as the number of GPU nodes and the pods are in the Running state. This means that ack-prometheus-gpu-exporter is deployed on the GPU nodes. If the pods are not in the Running state, run the `kubectl describe pod` command to query the reason why the pods are not running.

Step 4: Check whether data is collected by ack-prometheus-gpu-exporter

1. Run the following command to log on to a node in the cluster by using SSH:

```
ssh root@127.0.XX.XX
```

   - root: the custom account name.
   - 127.0.XX.XX: the public IP address of the node.

2. Run the following command to query the internal IP addresses of the pods:

```
kubectl get pods -n arms-prom -o wide
```

Expected output:

```
NAME                                READY   STATUS    RESTARTS   AGE    IP              NODE                        NOMINATED NODE   R
EADINESS GATES
ack-prometheus-gpu-exporter-4rdtl   1/1     Running   0          7h6m   172.21.XX.XX    cn-beijing.192.168.0.
22   <none>           <none>
ack-prometheus-gpu-exporter-vdkqf   1/1     Running   0          6d16h  172.21.XX.XX    cn-beijing.192.168.9
4.7  <none>           <none>
ack-prometheus-gpu-exporter-x7v48   1/1     Running   0          7h6m   172.21.XX.XX    cn-beijing.192.168.0
.23  <none>           <none>
```

3. Run the following command to call the `gpu exporter` service to obtain GPU metrics:

> **Note** By default, ack-prometheus-gpu-exporter uses port 9445.

```
curl 172.21.XX.XX:9445 | grep "nvidia_gpu"
```

Expected output:

```
 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  7518  100  7518    0     0   101k      0 --:--:-- --:--:-- --:--:--  101k
# HELP nvidia_gpu_duty_cycle Percent of time over the past sample period during which one or more kernels were executing on the GPU device
# TYPE nvidia_gpu_duty_cycle gauge
nvidia_gpu_duty_cycle{allocate_mode="exclusive",container_name="tfserving-gpu",minor_number="0",name="Tesla T4",namespace_name="default",node_name="cn-beijing.192.168.0.22",pod_name="fashion-mnist-eci-2-predictor-0-tfserving-proxy-tfserving-v789b",uuid="GPU-293f6608-281a-cc66-fcb3-0d366f32a31d"} 0
# HELP nvidia_gpu_memory_total_bytes Total memory of the GPU device
# TYPE nvidia_gpu_memory_total_bytes gauge
nvidia_gpu_memory_total_bytes{allocate_mode="exclusive",container_name="tfserving-gpu",minor_number="0",name="Tesla T4",namespace_name="default",node_name="cn-beijing.192.168.0.22",pod_name="fashion-mnist-eci-2-predictor-0-tfserving-proxy-tfserving-v789b",uuid="GPU-293f6608-281a-cc66-fcb3-0d366f32a31d"} 1.5811477504e+10
# HELP nvidia_gpu_memory_used_bytes Memory used by the GPU device
# TYPE nvidia_gpu_memory_used_bytes gauge
nvidia_gpu_memory_used_bytes{allocate_mode="exclusive",container_name="tfserving-gpu",minor_number="0",name="Tesla T4",namespace_name="default",node_name="cn-beijing.192.168.0.22",pod_name="fashion-mnist-eci-2-predictor-0-tfserving-proxy-tfserving-v789b",uuid="GPU-293f6608-281a-cc66-fcb3-0d366f32a31d"} 1.488453632e+10
# HELP nvidia_gpu_num_devices Number of GPU devices
# TYPE nvidia_gpu_num_devices gauge
nvidia_gpu_num_devices{node_name="cn-beijing.192.168.0.22"} 1
# HELP nvidia_gpu_power_usage_milliwatts Power usage of the GPU device in watts
# TYPE nvidia_gpu_power_usage_milliwatts gauge
nvidia_gpu_power_usage_milliwatts{allocate_mode="exclusive",container_name="tfserving-gpu",minor_number="0",name="Tesla T4",namespace_name="default",node_name="cn-beijing.192.168.0.22",pod_name="fashion-mnist-eci-2-predictor-0-tfserving-proxy-tfserving-v789b",uuid="GPU-293f6608-281a-cc66-fcb3-0d366f32a31d"} 27000
# HELP nvidia_gpu_temperature_celsius Temperature of the GPU device in celsius
# TYPE nvidia_gpu_temperature_celsius gauge
nvidia_gpu_temperature_celsius{allocate_mode="exclusive",container_name="tfserving-gpu",minor_number="0",name="Tesla T4",namespace_name="default",node_name="cn-beijing.192.168.0.22",pod_name="fashion-mnist-eci-2-predictor-0-tfserving-proxy-tfserving-v789b",uuid="GPU-293f6608-281a-cc66-fcb3-0d366f32a31d"} 44
```

If the output contains metric records that start with `nvidia_gpu`, data is collected by ack-prometheus-gpu-exporter.

16.4.4. Use a node pool to create a node with a custom NVIDIA driver version

By default, Container Service for Kubernetes (ACK) is installed with the NVIDIA driver of version 418.87.01. If your CUDA Toolkit requires a higher version of the NVIDIA driver, you must specify a custom NVIDIA driver version. This topic describes how to use a node pool to create a node with a custom NVIDIA driver version.

Benefits

This solution allows you to manage the NVIDIA drivers of different nodes in batches. The following two scenarios are covered:

- Node Pool A consists of all nodes on which you want to install the NVIDIA driver of version 418.181.07. If you want to schedule a task to a node that runs the NVIDIA driver of version 418.181.07, you need only to set the **selector** of the task to the label of Node Pool A.

- You manage cluster nodes in two groups: A and B. You want to install the NVIDIA driver of version 418.181.07 in Group A and the NVIDIA driver of version 450.102.0 in Group B. In this case, you can add nodes in Group A to Node Pool A and nodes in Group B to Node Pool B.

> **Note**
> - To upgrade the NVIDIA driver for a node, you must remove the node from its cluster and add the node back to the cluster. When the node is added to the cluster, the operating system of the node is reinstalled and the NVIDIA driver of the specified version is installed. Before you perform the upgrade, make sure that no task is running on the node and critical data is backed up.
> - To lower risk of failures, we recommend that you first upgrade the NVIDIA driver for one node. If no error occurs during this process, you can then perform the upgrade on multiple nodes.

Step 1: Determine the NVIDIA driver version

You must consider which CUDA Toolkit versions are compatible with the NVIDIA driver version that you want to use. The following figure displays a list of CUDA Toolkit versions and the compatible NVIDIA driver versions.

> **Note** By default, ACK clusters are installed with the NVIDIA driver of version 418.87.01.

CUDA Toolkit	Linux x86_64 Driver Version
CUDA 11.1 (11.1.0)	>= 450.80.02
CUDA 11.0 (11.0.3)	>= 450.36.06
CUDA 10.2 (10.2.89)	>= 440.33
CUDA 10.1 (10.1.105)	>= 418.39
CUDA 10.0 (10.0.130)	>= 410.48
CUDA 9.2 (9.2.88)	>= 396.26
CUDA 9.1 (9.1.85)	>= 390.46
CUDA 9.0 (9.0.76)	>= 384.81
CUDA 8.0 (8.0.61 GA2)	>= 375.26
CUDA 8.0 (8.0.44)	>= 367.48
CUDA 7.5 (7.5.16)	>= 352.31
CUDA 7.0 (7.0.28)	>= 346.46

For more information, see CUDA compatibility.

Step 2: Create a node pool and specify the NVIDIA driver version

Method 1: Select an NVIDIA driver version provided by ACK

> Note

The following example shows how to set the NVIDIA driver version to 418.181.07:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. Click **Create Node Pool** in the upper-right corner.
6. In the **Create Node Pool** dialog box, configure the parameters based on your requirements. For more information about the parameters, see Create a dedicated Kubernetes cluster. The following table describes the parameters:
   i. Click **Show Advanced Options**.
   ii. In the **Node Label** section, click  , set **Key** to `ack.aliyun.com/nvidia-driver-version`, and set **Value** to `418.181.07`.
   ACK provides the following NVIDIA driver versions:
   - 418.181.07
   - 450.102.04
   - 460.32.03

   iii. After you configure the parameters, click **Confirm Order**.

Method 2: Use a custom NVIDIA driver version

Custom driver version

The following example shows how to upload the *NVIDIA-Linux-x86_64-460.32.03.run* driver to specify a custom driver version. You can download NVIDIA drivers from the NVIDIA official website.

> **Note** The *NVIDIA-Linux-x86_64-460.32.03.run* file must be stored in the root directory of the Object Storage Service (OSS) bucket.

1. Create a bucket in the OSS console For more information, see Create buckets.
2. Upload the *NVIDIA-Linux-x86_64-460.32.03.run* file to the bucket. For more information, see Upload objects.
3. After the file is uploaded to the bucket, click **Files** in the left-side navigation pane of the bucket details page.
4. On the **Files** page, find the file that you uploaded and click **View Details** in the **Actions** column.
5. In the **View Details** panel, click **HTTPS** to disable HTTPS.

   > **Note** ACK downloads the driver file by using its URL, which adopts the HTTP protocol. By default, OSS uses the HTTPS protocol. You must click **HTTPS** to disable HTTPS.

6. Check the URL of the driver file. Take note of the URL. The URL consists of the following parts: endpoint

and runfile. For example, you can divide the following URL into two parts: `http://nvidia-XXX-XXX-cn-beijing.aliyuncs.com/NVIDIA-Linux-x86_64-460.32.03.run`.

- endpoint: nvidia-XXX-XXX-cn-beijing.aliyuncs.com
- runfile: NVIDIA-Linux-x86_64-460.32.03.run

Create a node pool with the custom driver version

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. Click **Create Node Pool** in the upper-right corner.
6. In the **Create Node Pool** dialog box, configure the parameters based on your requirements. For more information about the parameters, see Create a dedicated Kubernetes cluster. The following table describes the parameters:
   i. Click **Show Advanced Options**.
   ii. In the **Node Label** section, click  twice to add the following two labels:
      - For the first label, set **Key** to `ack.aliyun.com/nvidia-driver-oss-endpoint` and **Value** to `nvidia-XXX-XXX-cn-beijing.aliyuncs.com`.
      - For the second label, set **Key** to `ack.aliyun.com/nvidia-driver-runfile` and **Value** to `NVIDIA-Linux-x86_64-460.32.03.run`.
   iii. After you configure the parameters, click **Confirm Order**.

Step 3: Check whether the custom NVIDIA driver version is installed

1. In the ACK console, select the cluster to which the node belongs and click **More > Open Cloud Shell** in the **Actions** column.
2. Run the following command to query pods that have the `component: nvidia-device-plugin` label:

   ```
   kubectl get po -n kube-system -l component=nvidia-device-plugin -o wide
   ```

   Expected output:

   ```
   NAME                                                READY  STATUS   RESTARTS  AGE  IP             NODE                 NOMINATED NODE  READINESS GATES
   nvidia-device-plugin-cn-beijing.192.168.1.127  1/1   Running  0     6d   192.168.1.127  cn-beijing.192.168.1.127  <none>      <none>
   nvidia-device-plugin-cn-beijing.192.168.1.128  1/1   Running  0     17m  192.168.1.128  cn-beijing.192.168.1.128  <none>      <none>
   nvidia-device-plugin-cn-beijing.192.168.8.12   1/1   Running  0     9d   192.168.8.12   cn-beijing.192.168.8.12   <none>      <none>
   nvidia-device-plugin-cn-beijing.192.168.8.13   1/1   Running  0     9d   192.168.8.13   cn-beijing.192.168.8.13   <none>      <none>
   nvidia-device-plugin-cn-beijing.192.168.8.14   1/1   Running  0     9d   192.168.8.14   cn-beijing.192.168.8.14   <none>      <none>
   ```

   You can refer to the Node column to find the node that is newly added to the cluster. The name of the pod that runs on the node is `nvidia-device-plugin-cn-beijing.192.168.1.128`.
3. Run the following command to query the NVIDIA driver version of the node:

```
kubectl exec -ti nvidia-device-plugin-cn-beijing.192.168.1.128 -n kube-system -- nvidia-smi
```

Expected output:

```
Sun Feb  7 04:09:01 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.181.07   Driver Version: 418.181.07   CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:07.0 Off |                    0 |
| N/A   27C    P0    40W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000000:00:08.0 Off |                    0 |
| N/A   27C    P0    40W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  On   | 00000000:00:09.0 Off |                    0 |
| N/A   31C    P0    39W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  On   | 00000000:00:0A.0 Off |                    0 |
| N/A   27C    P0    41W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

The output shows that the NVIDIA driver version is 418.181.07, which indicates that the specified NVIDIA driver is installed.

Related information

- Use a node pool to upgrade the NVIDIA driver for a node

16.4.5. Use a node pool to upgrade the NVIDIA driver for a node

If your CUDA Toolkit requires a later version of the NVIDIA driver, you must upgrade the NVIDIA driver. To upgrade the NVIDIA driver for a node, you can remove the node from its cluster and add the node back to the cluster. When the node is added to the cluster, the operating system of the node is reinstalled. This allows you to install a specific version of the NVIDIA driver. This topic describes how to use a node pool to upgrade the NVIDIA driver for a node.

Context

Container Service for Kubernetes (ACK) does not allow you to upgrade the NVIDIA driver for a node without removing the node from its cluster. The node pool to which the node belongs may contain different nodes. Therefore, you cannot upgrade the NVIDIA driver for the entire node pool.

Benefits of using a node pool to upgrade the NVIDIA driver for a node

This solution allows you to manage the NVIDIA drivers of different nodes in batches. The following two scenarios are covered:

- You manage cluster nodes in two groups: A and B. You want to upgrade the NVIDIA driver of Group A to

version 418.181.07 and the NVIDIA driver of Group B to version 450.102.0. In this case, you can add nodes in Group A to Node Pool A and nodes in Group B to Node Pool B.

- Node Pool A consists of all nodes whose NVIDIA drivers are to be upgraded to version 418.181.07. If you want to schedule a task to a node that runs the NVIDIA driver of version 418.181.07, you need only to specify the **selector** of the task to the label of Node Pool A.

> **Note**
> - To upgrade the NVIDIA driver for a node, you must remove the node from its cluster and add the node back to the cluster. When the node is added to the cluster, the operating system of the node is reinstalled and the NVIDIA driver of the specified version is installed. Before you perform the upgrade, make sure that no task is running on the node and critical data is backed up.
> - To lower risk of failures, we recommend that you first upgrade the NVIDIA driver for one node. If no error occurs during this process, you can then perform the upgrade on multiple nodes.

Step 1: Determine the NVIDIA driver version

You must consider which CUDA Toolkit versions are compatible with the NVIDIA driver version that you want to use. The following figure displays a list of CUDA Toolkit versions and the compatible NVIDIA driver versions.

> **Note** By default, ACK clusters are installed with the NVIDIA driver of version 418.87.01.

CUDA Toolkit	Linux x86_64 Driver Version
CUDA 11.1 (11.1.0)	>= 450.80.02
CUDA 11.0 (11.0.3)	>= 450.36.06
CUDA 10.2 (10.2.89)	>= 440.33
CUDA 10.1 (10.1.105)	>= 418.39
CUDA 10.0 (10.0.130)	>= 410.48
CUDA 9.2 (9.2.88)	>= 396.26
CUDA 9.1 (9.1.85)	>= 390.46
CUDA 9.0 (9.0.76)	>= 384.81
CUDA 8.0 (8.0.61 GA2)	>= 375.26
CUDA 8.0 (8.0.44)	>= 367.48
CUDA 7.5 (7.5.16)	>= 352.31
CUDA 7.0 (7.0.28)	>= 346.46

For more information, see CUDA compatibility.

Step 2: Remove a node from its cluster

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. Select the node for which you want to upgrade the NVIDIA driver and click **Batch Remove**.
6. In the **Remove Node** dialog box, select **Drain the Node** and click **OK**.

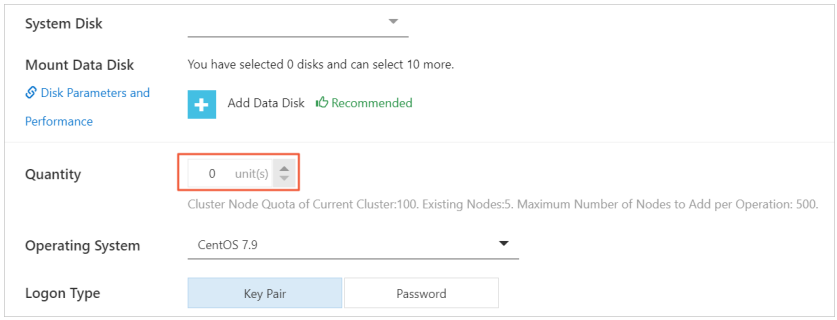
Step 3: Create a node pool and specify the NVIDIA driver version

Method 1: Select an NVIDIA driver version provided by ACK

> **Note** This method is simple. You need only to add the `ack.aliyun.com/nvidia-driver-version=<Driver Version>` label when you create a node pool and then add the node that you removed to the node pool.

The following example shows how to set the NVIDIA driver version to 418.181.07:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. In the upper-right corner of the Node Pools page, click **Create Node Pool**.
6. In the **Create Node Pool** dialog box, set the required parameters. For more information about the parameters, see Create a dedicated Kubernetes cluster. The following table describes the parameters.

Parameter	Description
VSwitch	The VSwitch and Instance Type parameters are used only when new nodes are added to the node pool. In this example, an existing node is added to the node pool. Therefore, you can set the parameters to any values.
Instance Type	
Operating System	The operating system of the node after the node is added to the cluster.
Quantity	Set the value to 0. Otherwise, ACK creates new instances. 

   i. Click **Show Advanced Options**.

ii. In the **Node Label** section, click  , set **Key** to `ack.aliyun.com/nvidia-driver-version`, and then set **Value** to `418.181.07`.

ACK provides the following NVIDIA driver versions:

- 418.181.07
- 450.102.04
- 460.32.03

iii. Click **Confirm Order**.

Method 2: Use a custom NVIDIA driver version

Custom driver version

The following example shows how to upload the *NVIDIA-Linux-x86_64-460.32.03.run* file to specify a custom driver version. You can download NVIDIA driver files from the NVIDIA official website.

> **Note** The *NVIDIA-Linux-x86_64-460.32.03.run* file must be stored in the root directory of the specified Object Storage Service (OSS) bucket.

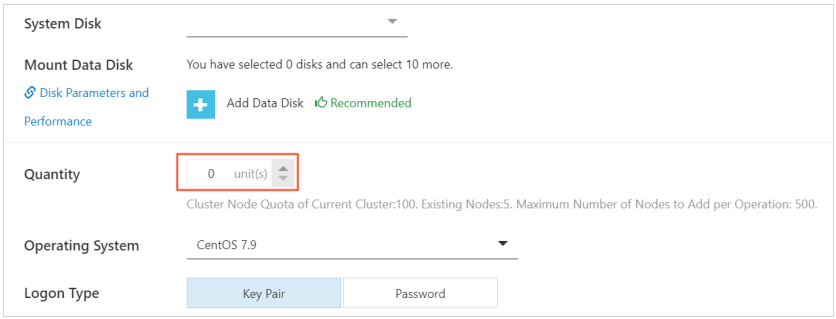
1. Create an OSS bucket in the OSS console. For more information, see Create buckets.
2. Upload the *NVIDIA-Linux-x86_64-460.32.03.run* file to the OSS bucket. For more information, see Upload objects.
3. After the file is uploaded to the bucket, click **Files** in the left-side navigation pane of the bucket details page.
4. On the **Files** page, find the file that you uploaded and click **View Details** in the **Actions** column.
5. In the **View Details** panel, turn off the **HTTPS** switch to disable HTTPS.

   > **Note** ACK downloads the driver file by using its URL, which adopts the HTTP protocol. By default, OSS uses the HTTPS protocol. You must turn off the **HTTPS** switch to disable HTTPS.

6. Check the URL of the driver file. Take note of the URL. The URL consists of the following parts: endpoint and runfile. For example, you can divide the following URL into two parts: `http://nvidia-XXX-XXX-cn-beijing.aliyuncs.com/NVIDIA-Linux-x86_64-460.32.03.run`.
   - endpoint: nvidia-XXX-XXX-cn-beijing.aliyuncs.com
   - runfile: NVIDIA-Linux-x86_64-460.32.03.run

Create a node pool with the custom driver version

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. In the upper-right corner of the Node Pools page, click **Create Node Pool**.
6. In the **Create Node Pool** dialog box, configure the required parameters. For more information about the parameters, see Create a dedicated Kubernetes cluster. The following table describes the parameters.

Parameter	Description
VSwitch	The VSwitch and Instance Type parameters are used only when new nodes are added to the node pool. In this example, an existing node is added to the node pool. Therefore, you can set the parameters to any values.
Instance Type	
Operating System	The operating system of the node after the node is added to the cluster.
Quantity	Set the value to 0. Otherwise, ACK creates new instances. 

i. Click **Show Advanced Options**.

ii. In the **Node Label** section, click  twice to add the following two labels:

- For the first label, set **Key** to `ack.aliyun.com/nvidia-driver-oss-endpoint` and **Value** to `nvidia-XXX-XXX-cn-beijing.aliyuncs.com`.
- For the second label, set **Key** to `ack.aliyun.com/nvidia-driver-runfile` and **Value** to `NVIDIA-Linux-x86_64-460.32.03.run`.

iii. Click **Confirm Order**.

Step 4: Add the node to the node pool

After the node pool is created, add the node that you removed to the node pool. For more information, see Add existing ECS instances to an ACK cluster.

Step 5: Check whether the NVIDIA driver version is upgraded

1. In the ACK console, select the cluster to which the node belongs and choose **More > Open Cloud Shell** in the **Actions** column.
2. Run the following command to query pods that have the `component: nvidia-device-plugin` label:

```
kubectl get po -n kube-system -l component=nvidia-device-plugin -o wide
```

Expected output:

```
NAME                                                READY   STATUS    RESTARTS   AGE   IP              NODE                 NOMINATED NODE   REA
DINESS GATES
nvidia-device-plugin-cn-beijing.192.168.1.127   1/1     Running   0          6d    192.168.1.127   cn-beijing.192.168.1.1
27   <none>           <none>
nvidia-device-plugin-cn-beijing.192.168.1.128   1/1     Running   0          17m   192.168.1.128   cn-beijing.192.168.1.
128   <none>           <none>
nvidia-device-plugin-cn-beijing.192.168.8.12    1/1     Running   0          9d    192.168.8.12    cn-beijing.192.168.8.12
<none>           <none>
nvidia-device-plugin-cn-beijing.192.168.8.13    1/1     Running   0          9d    192.168.8.13    cn-beijing.192.168.8.13
<none>           <none>
nvidia-device-plugin-cn-beijing.192.168.8.14    1/1     Running   0          9d    192.168.8.14    cn-beijing.192.168.8.14
<none>           <none>
```

You can refer to the NODE column to find the node that is newly added to the cluster. The name of the pod that runs on the node is `nvidia-device-plugin-cn-beijing.192.168.1.128`.

3. Run the following command to query the NVIDIA driver version of the node:

```
kubectl exec -ti nvidia-device-plugin-cn-beijing.192.168.1.128 -n kube-system -- nvidia-smi
```

Expected output:

```
Sun Feb  7 04:09:01 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.181.07   Driver Version: 418.181.07   CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:07.0 Off |                    0 |
| N/A   27C    P0    40W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000000:00:08.0 Off |                    0 |
| N/A   27C    P0    40W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  On   | 00000000:00:09.0 Off |                    0 |
| N/A   31C    P0    39W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  On   | 00000000:00:0A.0 Off |                    0 |
| N/A   27C    P0    41W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

The output shows that the NVIDIA driver version is 418.181.07. This indicates that the NVIDIA driver is upgraded.

Related information

- Use a node pool to create a node with a custom NVIDIA driver version

16.4.6. Manually upgrade the NVIDIA driver for a node

If your CUDA Toolkit requires a higher version of the NVIDIA driver, you must upgrade the NVIDIA driver. This topic describes how to manually upgrade the NVIDIA driver for a node.

Prerequisites

Connect to Kubernetes clusters by using kubectl

Procedure

1. Run the following command to set the GPU node that you want to upgrade to unschedulable:

   ```
   kubectl cordon <$Node_ID>
   ```

   > **Note** Replace *<$Node_ID>* with the ID of the node.

   Expected output:

   ```
   node/<$Node_ID> already cordoned
   ```

2. Run the following command to remove the GPU node that you want to upgrade from the cluster:

   ```
   kubectl drain <$Node_ID> --grace-period=120 --ignore-daemonsets=true
   ```

   Expected output:

   ```
   node/cn-beijing.<$Node_ID> cordoned
   WARNING: Ignoring DaemonSet-managed pods: flexvolume-9scb4, kube-flannel-ds-r2qmh, kube-proxy-worker-l62sf, logtail-ds-f9vbg
   pod/nginx-ingress-controller-78d847fb96-5fkkw evicted
   ```

3. Uninstall the NVIDIA driver from the GPU node.

   > **Note** In this step, the NVIDIA driver of version 418.87.01 is uninstalled. If the version of the NVIDIA driver that you want to uninstall is not 418.87.01, go to the NVIDIA official website and download the installation package of the NVIDIA driver. You must replace the installation package of version 418.87.01 with your downloaded installation package.

   i. Log on to the GPU node and run the following command to query the NVIDIA driver version.

   > **Note** For more information about how to log on to a GPU node, see Connect to a Linux instance by using password authentication and Connect to a Windows instance by using password authentication.

   ```
   nvidia-smi -a | grep 'Driver Version'
   ```

   Expected output:

   ```
   Driver Version                  : 418.87.01
   ```

ii. Run the following command to download the installation package of the NVIDIA driver that you want to uninstall:

```
cd /tmp/
wget https://us.download.nvidia.com/tesla/418.87/NVIDIA-Linux-x86_64-418.87.01.run
```

iii. Run the following command to uninstall the current NVIDIA driver:

```
chmod u+x NVIDIA-Linux-x86_64-418.87.01.run
./NVIDIA-Linux-x86_64-418.87.01.run --uninstall -a -s -q
```

4. Run the following command to restart the GPU node:

```
reboot
```

5. Run the following command to download and install the NVIDIA driver version to which you want to upgrade. In this example, version 418.181.07 is used.

```
cd /tmp/
wget https://us.download.nvidia.com/tesla/418.181.07/NVIDIA-Linux-x86_64-418.181.07.run
chmod u+x NVIDIA-Linux-x86_64-418.181.07.run
sh ./NVIDIA-Linux-x86_64-418.181.07.run -a -s -q
```

6. Refer to the following commands to configure the parameters based on your business requirements:

```
nvidia-smi -pm 1 || true
nvidia-smi -acp 0 || true
nvidia-smi --auto-boost-default=0 || true
nvidia-smi --auto-boost-permission=0 || true
nvidia-modprobe -u -c=0 -m || true
```

7. To start the NVIDIA driver at startup, check the */etc/rc.d/rc.local* file and make sure that the following content is added to the file.

```
nvidia-smi -pm 1 || truenvidia-smi -acp 0 || truenvidia-smi --auto-boost-default=0 || truenvidia-smi --auto-boo
st-permission=0 || truenvidia-modprobe -u -c=0 -m || true
```

8. Run the following command to restart kubelet and the Docker runtime:

```
systemctl restart docker && systemctl restart kubelet
```

9. Run the following command to set the GPU node to schedulable:

```
kubectl uncordon <$Node_ID>
```

Expected output:

```
node/<$Node_ID> already uncordoned
```

10. Run the following command on the device-plugin pod of the GPU node to check the driver version:

```
kubectl exec -n kube-system -t nvidia-device-plugin-<$Node_ID> nvidia-smi
```

Expected output:

```
Thu Jan 17 00:33:27 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.181.07   Driver Version: 418.181.07   CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  On   | 00000000:00:09.0 Off |                    0 |
| N/A   27C    P0    28W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------
```

The preceding output indicates that the NVIDIA driver is upgraded.
If the NVIDIA driver is not upgraded, you can run the `docker ps` command to check whether containers are started on the GPU node. For more information about how to troubleshoot container startup issues, see Failed to start a container on the GPU node.

16.4.7. Fix the issue that the IDs of GPUs are changed after a GPU-accelerated ECS instance is restarted or replaced

After a GPU-accelerated instance fails, the IDs of the GPUs on the instance may be changed. If the GPU IDs are changed, the containers cannot be launched as normal. GPUOps is used to detect whether the IDs of the GPUs on a GPU-accelerated instance are the same as those stored in the */var/lib/kubelet/device-plugins/kubelet_internal_checkpoint* file. If the GPU IDs are not the same, GPUOps deletes the `checkpoint` file. Then, Kubelet generates another `checkpoint` file. This ensures that the GPU IDs stored in the checkpoint file are the same as the actual GPU IDs. This topic describes how to fix the issue that the IDs of the GPUs on a GPU-accelerated instance are changed after the instance fails.

Prerequisites

- Create a managed Kubernetes cluster with GPU-accelerated nodes or Create a dedicated Kubernetes cluster with GPU-accelerated nodes.
- A jump server that uses SSH is added to the cluster. The jump server has access to the Internet. For more information, see Configure SNAT entries for existing ACK clusters.

Context

- After a failed GPU-accelerated instance is restarted or replaced, the IDs of the GPUs on the instance may be changed. If the GPU IDs are different from those stored in the */var/lib/kubelet/device-plugins/kubelet_internal_checkpoint* file, containers on the GPU-accelerated instance cannot be launched as normal.

  > **Note** The ID of a GPU may be changed in the following scenarios:
  > - A failed GPU-accelerated instance is restarted or replaced.
  > - You manually restart a GPU-accelerated instance.

- GPUOps is a binary program that can run on Linux. For more information about how to download GPUOps,

see GPUOps.

- A GPU-accelerated instance can be equipped with multiple GPUs. Each GPU has a unique ID.

Step 1: Deploy GPUOps

Deploy GPUOps on a single node

1. Run the following command to copy GPUOps to the */usr/local/bin/* directory and grant executable permissions to GPUOps:

```
cp ./gpuops /usr/local/bin/
chmod +x gpuops
```

2. Run the following command to enable GPUOps to automatically start up with the instance:

```
cat > /etc/systemd/system/gpuops.service <<EOF
[Unit]
Description=Gpuops: check kubelet checkpoint gpu status
After=network-online.target
Wants=network-online.target
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/bin/gpuops check
[Install]
WantedBy=multi-user.target
EOF
systemctl enable gpuops.service
```

Deploy GPUOps on multiple nodes at a time

In the Kubernetes cluster with GPU-accelerated instances, deploy a DaemonSet by using the following YAML template:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: gpuops-deploy
 labels:
  k8s-app: gpuops-deploy
spec:
 selector:
  matchLabels:
   name: gpuops-deploy
 template:
  metadata:
   labels:
    name: gpuops-deploy
  spec:
   hostPID: true
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
       - matchExpressions:
         - key: aliyun.accelerator/nvidia_name
           operator: Exists
   containers:
    - name: gpuops
      image: registry.cn-beijing.aliyuncs.com/acs/gpuops:latest
      command:
      securityContext:
       privileged: true
      volumeMounts:
       - name: hostbin
         mountPath: /workspace/host/usr/local/bin
       - name: hostsystem
         mountPath: /workspace/host/etc/systemd/system
   terminationGracePeriodSeconds: 30
   volumes:
    - name: hostbin
      hostPath:
       path: /usr/local/bin
    - name: hostsystem
      hostPath:
       path: /etc/systemd/system
```

After the DaemonSet is deployed, the Kubernetes cluster with GPU-accelerated instances performs the following operations on all of the worker nodes:

- Copies GPUOps to the */usr/local/bin/* directory and makes GPUOps executable.
- Enables GPUOps to automatically start up with the instance.

When a failed GPU-accelerated instance in the cluster is restarted or replaced, GPUOps ensures that the GPU IDs stored in the checkpoint file are the same as the actual GPU IDs.

Step 2: Verify that GPUOps has fixed the GPU ID issue

1. After GPUOps is deployed, create a pod in the Kubernetes cluster with GPU-accelerated instances.

   The following YAML template is used to create the pod:

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: app-3g-v1
 labels:
  app: app-3g-v1
spec:
 replicas: 1
 serviceName: "app-3g-v1"
 podManagementPolicy: "Parallel"
 selector: # define how the deployment finds the pods it manages
  matchLabels:
   app: app-3g-v1
 template: # define the pods specifications
  metadata:
   labels:
    app: app-3g-v1
  spec:
   containers:
   - name: app-3g-v1
    image: registry.cn-shanghai.aliyuncs.com/tensorflow-samples/cuda-malloc:3G
    resources:
     limits:
      nvidia.com/gpu: 1
```

2. Run the following command to query the status of the pod:

```
kubectl get pod
```

Expected output:

```
NAME              READY  STATUS   RESTARTS  AGE
app-3g-v1-0          1/1  Running  0     22s
```

3. Log on to the GPU-accelerated instance where the pod runs and run the following command to restart the instance.

> **Note** For more information about how to log on to a GPU-accelerated instance, see Connect to a Linux instance by using password authentication.

```
sudo reboot
```

4. Run the following command to print the log of GPUOps:

```
journalctl -u gpuops
```

- If the following output is returned, the GPU ID stored in the `checkpoint` file is the same as the actual GPU ID. You do not need to perform any actions.

```
June 28 14:49:00 iZ2vc1mysgx8bqdv3oyji9Z gpuops[21976]: {"level":"info","msg":"check gpu start...","tim
e":"2020-06-28T14:49:00+08:00"}
June 28 14:49:00 iZ2vc1mysgx8bqdv3oyji9Z gpuops[21976]: {"level":"info","msg":"find nvidia gpu: [\"GPU-
383cd6a5-00a6-b455-62c4-c163d164b837\"]","time":"2020-06-28T14:49:00+08:00"}
June 28 14:49:00 iZ2vc1mysgx8bqdv3oyji9Z gpuops[21976]: {"level":"info","msg":"read checkpoint: {\"Dat
a\":{\"PodDeviceEntries\":[{\"PodUID\":\"300f47eb-e5c3-4d0b-9a87-925837b67732\",\"ContainerName\":\"
app-3g-v1\",\"ResourceName\":\"aliyun.com/g
June 28 14:49:00 iZ2vc1mysgx8bqdv3oyji9Z systemd[1]: Started Gpuops: check kubelet checkpoint gpu st
atus.
June 28 14:49:00 iZ2vc1mysgx8bqdv3oyji9Z gpuops[21976]: {"level":"info","msg":"find registered gpu: [\"
GPU-383cd6a5-00a6-b455-62c4-c163d164b837\"]","time":"2020-06-28T14:49:00+08:00"}
June 28 14:49:00 iZ2vc1mysgx8bqdv3oyji9Z gpuops[21976]: {"level":"info","msg":"find pod gpu: null","tim
e":"2020-06-28T14:49:00+08:00"}
June 28 14:49:00 iZ2vc1mysgx8bqdv3oyji9Z gpuops[21976]: {"level":"info","msg":"cached gpu info in chec
kpoint is up to date, check gpu finished","time":"2020-06-28T14:49:00+08:00"}
```

- If the output contains `warning` messages, the GPU ID is changed. In this case, GPUOps deletes the `checkpoint` file and Kubernetes generates another `checkpoint` file to store the new GPU ID.

```
June 28 14:41:16 iZ2vc1mysgx8bqdv3oyji9Z systemd[1]: Starting Gpuops: check kubelet checkpoint gpu st
atus...
June 28 14:41:16 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"info","msg":"check gpu start...","time":
"2020-06-28T14:41:16+08:00"}
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"info","msg":"find nvidia gpu: [\"GPU-68
ce64fb-ea68-5ad6-7e72-31f3f07378df\"]","time":"2020-06-28T14:41:17+08:00"}
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"info","msg":"read checkpoint: {\"Data\"
:{\"PodDeviceEntries\":[{\"PodUID\":\"300f47eb-e5c3-4d0b-9a87-925837b67732\",\"ContainerName\":\"ap
p-3g-v1\",\"ResourceName\":\"aliyun.com/gpu
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"info","msg":"find registered gpu: [\"GP
U-7fd52bfc-364d-1129-a142-c3f10e053ccb\"]","time":"2020-06-28T14:41:17+08:00"}
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"info","msg":"find pod gpu: [\"GPU-7fd5
2bfc-364d-1129-a142-c3f10e053ccb\"]","time":"2020-06-28T14:41:17+08:00"}
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"warning","msg":"the registered gpu uu
id not equal with nvidia gpu","time":"2020-06-28T14:41:17+08:00"}
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"warning","msg":"cached gpu info in ch
eckpoint is out of date","time":"2020-06-28T14:41:17+08:00"}
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"info","msg":"delete checkpoint file suc
cess","time":"2020-06-28T14:41:17+08:00"}
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"info","msg":"kubelet restart success","
time":"2020-06-28T14:41:17+08:00"}
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z gpuops[951]: {"level":"info","msg":"check gpu finished","time
":"2020-06-28T14:41:17+08:00"}
June 28 14:41:17 iZ2vc1mysgx8bqdv3oyji9Z systemd[1]: Started Gpuops: check kubelet checkpoint gpu st
atus.
```

  - Run the following command to query the actual GPU ID:

    ```
    nvidia-smi -L
    ```

    Expected output:

    ```
    GPU 0: Tesla T4 (UUID: GPU-0650a168-e770-3ea8-8ac3-8a1d419763e0)
    ```

- Run the following command to query the GPU ID stored in the `checkpoint` file:

  ```
  cat /var/lib/kubelet/device-plugins/kubelet_internal_checkpoint
  ```

  Expected output:

  ```
  {"Data":{"PodDeviceEntries":null,"RegisteredDevices":{"nvidia.com/gpu":["GPU-0650a168-e770-3ea8-8ac3-8a1d419763e0"]}},"Checksum":3952659280}
  ```

  The preceding output shows that the ID stored in the `checkpoint` file is the same as the actual GPU ID. This indicates that GPUOps has fixed the issue of inconsistent GPU IDs.

16.4.8. Update the NVIDIA driver license of a vGPU-accelerated instance

Worker nodes can be deployed on vGPU-accelerated instances. A vGPU-accelerated instance is a lightweight GPU-accelerated instance. To use vGPU-accelerated instances, you must purchase NVIDIA driver licenses. Before you can update a license based on scenarios described in this topic, make sure that you have purchased an NVIDIA driver license.

> **Note**
> - If no license is purchased, click here to purchase a license.
> - The scenarios described in this topic are suitable to dedicated, managed, and professional Kubernetes clusters.

Scenario 1: An ACK cluster contains vGPU-accelerated instances

1. If your Container Service for Kubernetes (ACK) cluster contains vGPU-accelerated instances, log on to a vGPU-accelerated instance in your cluster and modify the following field settings in the */etc/nvidia/gridd.conf* file:

   > **Note** For more information about how to log on to a GPU-accelerated node, see Connect to a Linux instance by using password authentication and Connect to a Windows instance by using password authentication.

   ```
   ServerAddress=<your License Server address>
   ServerPort=<License Server port>
   ```

2. Run the following command to restart vGPU:

   ```
   systemctl daemon-reload
   systemctl restart nvidia-gridd
   ```

3. Run the following command to check whether the command is successfully run:

   ```
   nvidia-smi
   ```

   Expected output:

```
Wed Jun  2 14:22:29 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.102.04   Driver Version: 450.102.04   CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:00:07.0 Off |                    0 |
| N/A   28C    P8    8W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

The output indicates that the `nvidia-smi` command is successfully run. This means that the NVIDIA driver license of the vGPU-accelerated instance is updated.

Scenario 2: An ACK cluster does not contain vGPU-accelerated instances

If your ACK cluster does not contain vGPU-accelerated instances, you must first add a vGPU-accelerated instance to the cluster. To do this, you can manually add an existing vGPU-accelerated instance or enable the system to automatically add an existing vGPU-accelerated instance. Then, perform the steps as described in Scenario 1.

For more information about how to add an instance to an ACK cluster, see Add existing ECS instances to an ACK cluster.

16.5. NPU resource scheduling

16.5.1. Perform NPU scheduling in a Kubernetes cluster

This topic describes how to create and use a Container Service for Kubernetes (ACK) cluster with neural processing unit (NPU) resources.

Prerequisites

ACK and Resource Access Management (RAM) services are activated.

Context

Compared with CPUs, the most obvious advantage of NPUs is fast data processing in complex algorithmic models. Different from the von Neumann architecture used by CPUs, NPUs adopt the data-driven parallel computing architecture. This architecture significantly increases computing capabilities and reduces power consumption by using data streams. NPUs are the most suitable option in scenarios where large amounts of data are processed, such as video and image processing. Compared with CPUs, NPUs offer 100 to 1,000 times faster processing speeds and also significantly lower power consumption.

You can create ACK clusters and use Hanguang NPU to run compute-intensive tasks, such as machine learning and image processing. You can deploy workloads and dynamically scale resources to meet your business requirements.

> **Note** For more information about Hanguang NPU, visit this website.

This topic describes how to use NPU resources in a Kubernetes cluster. In the following example, an ACK cluster is created and an ecs.ebman1.26xlarge instance is added to the cluster.

ACK performs the following operation when it creates the cluster:

- Create Elastic Compute Service (ECS) instances, configures a public key to enable Secure Shell (SSH) logon from master nodes to other nodes, and then configures the ACK cluster by using cloud-init.
- Create a security group that allows access to a virtual private cloud (VPC) over Internet Control Message Protocol (ICMP).
- If you do not specify an existing VPC, ACK creates a VPC and a vSwitch and creates SNAT rules for the vSwitch.
- Add route entries to the VPC.
- Create a NAT gateway and an elastic IP address (EIP).
- Create a RAM user and grants it permissions to query, create, and delete ECS instances, and permissions to add and delete disks. The RAM user is also granted all permissions on Server Load Balancer (SLB), CloudMonitor, VPC, Log Service, and Apsara File Storage NAS. ACK also creates an AccessKey pair for the RAM user. ACK automatically creates SLB instances, disks, and VPC route entries based on your configurations.
- Create an internal-facing SLB instance and open port 6443.
- (Optional)Create an Internet-facing SLB instance and open ports 6443, 8443, and 22. If you enable SSH logon when you create the cluster, port 22 is open. Otherwise, port 22 is not open.

Use NPU resources

To allow a pod to use NPU resources, set the `aliyun.com/npu` parameter in `resources.limits`.

```
apiVersion: v1
kind: Pod
metadata:
 name: <The name of the pod>
spec:
 containers:
  - name: <The name of the container>
    image: <The name of the image>
    resources:
     limits:
      aliyun.com/npu: <The amount of NPU resources>
```

Set up an NPU environment to run TensorFLow

You can use NPU resources to train TensorFlow models. In the following example, a pod is started to perform model training with NPU resources.

1. Connect to the ACK cluster. For more information, see Use kubectl on Cloud Shell to manage ACK clusters.

   Run the following command in Cloud Shell:

```
cat > test-pod.yaml <<- EOF
apiVersion: v1
kind: Pod
metadata:
 name: test-npu-pod
spec:
 restartPolicy: Never
 imagePullSecrets:
  - name: regsecret
 containers:
  - name: resnet50-npu
   image: registry.cn-shanghai.aliyuncs.com/hgai/tensorflow:v1_resnet50-tensorflow1.9.0-toolchain1.0.2-centos7.6
    resources:
     limits:
      aliyun.com/npu: 1 # requesting  NPUs
EOF
```

2. Run the following command to create a pod:

```
kubectl apply -f test-pod.yaml
```

3. Run the following command to query the pod state:

```
kubectl get po test-npu-pod
```

> **Note** If the pod state is Error, run the `kubectl logs test-npu-pod` command to check the pod log and troubleshoot the error.

Result

Wait a few minutes and query the pod state again.

```
kubectl get po test-npu-pod
```

If the pod state is Completed, check the pod log again.

```
kubectl logs test-npu-pod
```

The following output indicates that the training is complete.

```
2019-10-30 12:10:50.389452: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 AVX512F FMA
100%|##########| 98/98 [00:26<00:00,  3.67it/s]
resnet_v1_50, result = {'top_5': 0.9244321584701538, 'top_1': 0.7480267286300659}
```

16.6. Use the MIG feature of NVIDIA A100 GPUs in an ACK cluster

An NVIDIA A100 GPU provides higher performance over a V100 GPU and adds a third-generation Tensor Core. The Multi-Instance GPU (MIG) feature partitions an NVIDIA A100 GPU into separate GPU instances of different sizes. This delivers guaranteed quality of service (QoS) and achieves fault isolation. MIG turns an NVIDIA A100 GPU into an elastic resource pool. You can dynamically schedule GPU instances of different sizes in the pool to workloads. This topic describes how to use the MIG feature of an NVIDIA A100 GPU in a Container Service for Kubernetes (ACK) cluster.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.

  > **Note** The MIG feature of NVIDIA A100 GPUs is supported by dedicated Kubernetes clusters, managed Kubernetes clusters, and professional Kubernetes clusters.

- The Kubernetes version of the ACK cluster must be 1.18.8 or later.
- The MIG feature has limits on zones and the stock of Elastic Compute Service (ECS) instances. If the **ecs.ebmgn7.26xlarge** instance type is unavailable when you create a node pool, Submit a ticket.
- A jump server that uses SSH is added to the cluster. The jump server has access to the Internet. For more information, see Configure SNAT entries for existing ACK clusters.

Context

NVIDIA A100 GPUs are designed for scientific computing and analytics of cloud graphs and data.

TensorCore

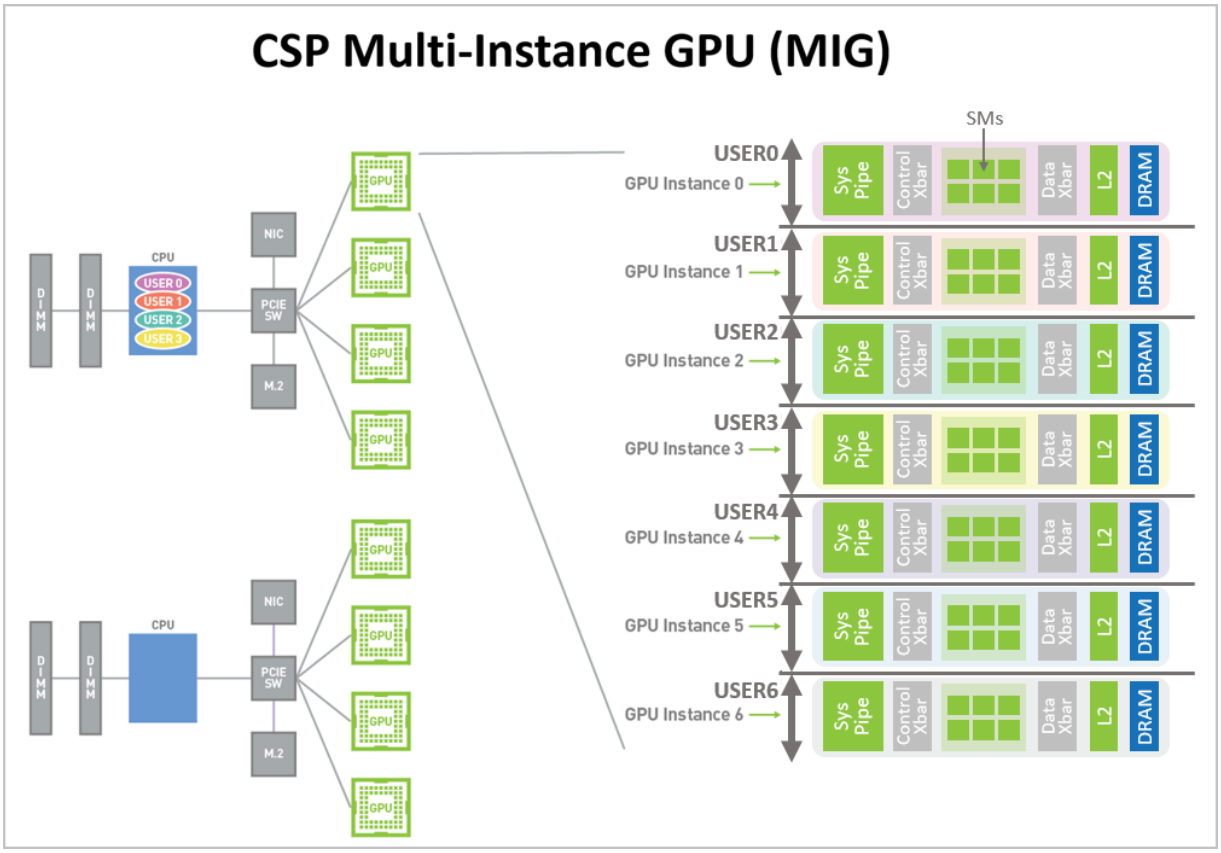
An NVIDIA A100 GPU includes 19.5 teraflops of FP32 performance, 6,912 CUDA cores, and 40 GiB of memory. Each streaming multiprocessor (SM) includes 64 FP32 CUDA cores. An A100 GPU provides higher performance over a V100 GPU in terms of the following aspects:

- The NVIDIA A100 GPU adds a third-generation Tensor Core. The throughput of sparse matrices for High Performance Computing (HPC) and deep learning on A100 GPUs is twice of that on V100 GPUs.
- The third-generation Tensor Core can accelerate the computing of the following data types: FP16, BF16, TF32, FP64, INT8, INT4, and Binary.
- An A100 GPU provides more options on precision. TF32 Tensor Core operations on an A100 GPU can accelerate FP32 input/output in deep learning frameworks and HPC. TF32 Tensor Core operations run 10 times faster than V100 FP32 FMA operations and 20 times faster with sparse matrices.
- For FP16/FP32 mixed-precision deep learning, A100 Tensor Core is 2.5 times faster than V100 and 5 times faster with sparse matrices.
- A100 sparse INT8 is 20 times faster than V100 INT8.
- For HPC, A100 Tensor Core is 2.5 times faster than the FP64 performance of V100.

MIG

You can use the virtualization and GPU partitioning capabilities of MIG to partition an individual A100 GPU into up to seven GPU instances without extra cost.

MIG can partition an A100 GPU into multiple GPU instances. The SMs of each GPU instance have separate and isolated paths through the entire memory system. The on-chip crossbar ports, L2 cache banks, memory controllers, and DRAM address buses are all uniquely allocated to an individual GPU instance. This ensures that the workload of an individual user can run as normal. Even if the L2 caches and DRAM interfaces of one GPU instance are overloaded, other GPU instances are not affected. Each GPU instance is provided with guaranteed QoS and fault isolation.



Step 1: Create an ECS instance that is equipped with A100 GPUs

ACK allows you to add GPU-accelerated ECS instances that support MIG. You can purchase or add existing ECS instances equipped with A100 GPUs to an ACK cluster in the ACK console.

Log on to the Container Service for Kubernetes (ACK) console. Find the cluster that you want to manage and create a node pool for the cluster. In the **Create Node Pool** dialog box, select **ecs.ebmgn7.26xlarge** in the **Instance Type** section.

For more information, see 管理节点池.

Step 2: Configure MIG

1. Log on to the GPU-accelerated ECS instance and run the following command to query information about the GPU:

   ```
   nvidia-smi
   ```

   > **Note** For more information about how to log on to a GPU-accelerated ECS instance, see Connect to a Linux instance by using password authentication or Connect to a Windows instance by using password authentication.

   Expected output:

```
[root@iZ[redacted]Z ~]# nvidia-smi
Fri Mar  5 20:29:43 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.102.04   Driver Version: 450.102.04   CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  A100-SXM4-40GB      On   | 00000000:52:00.0 Off |                    On |
| N/A   29C    P0    69W / 350W |      0MiB / 40537MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+
|   1  A100-SXM4-40GB      On   | 00000000:57:00.0 Off |                    On |
| N/A   30C    P0    66W / 350W |      0MiB / 40537MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+
|   2  A100-SXM4-40GB      On   | 00000000:6D:00.0 Off |                    On |
| N/A   29C    P0    65W / 350W |      0MiB / 40537MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+
|   3  A100-SXM4-40GB      On   | 00000000:73:00.0 Off |                    On |
| N/A   29C    P0    67W / 350W |      0MiB / 40537MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+
|   4  A100-SXM4-40GB      On   | 00000000:92:00.0 Off |                    On |
| N/A   30C    P0    67W / 350W |      0MiB / 40537MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+
|   5  A100-SXM4-40GB      On   | 00000000:97:00.0 Off |                    On |
| N/A   28C    P0    68W / 350W |      0MiB / 40537MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+
|   6  A100-SXM4-40GB      On   | 00000000:AD:00.0 Off |                    On |
| N/A   29C    P0    66W / 350W |      0MiB / 40537MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+
|   7  A100-SXM4-40GB      On   | 00000000:B3:00.0 Off |                    On |
| N/A   30C    P0    67W / 350W |      0MiB / 40537MiB |     N/A      Default |
|                               |                      |              Enabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| MIG devices:                                                                |
```

The preceding figure shows that the ECS instance is equipped with eight A100 GPUs. Each A100 GPU runs independently and MIG is disabled for all GPUs.

2. Run the following command to view the GPU partition sizes that are supported by a specified A100 GPU:

```
nvidia-smi mig -i 0 -lgip
```

Expected output:

```
[root@iz[redacted] ~]# nvidia-smi mig -i 0 -lgip
+-----------------------------------------------------------------------------+
| GPU instance profiles:                                                      |
| GPU   Name          ID    Instances   Memory     P2P    SM    DEC   ENC   |
|                           Free/Total    GiB              CE    JPEG  OFA   |
|=============================================================================|
|   0  MIG 1g.5gb     19     7/7         4.75       No     14     0     0   |
|                                                          1     0     0   |
+-----------------------------------------------------------------------------+
|   0  MIG 2g.10gb    14     3/3         9.75       No     28     1     0   |
|                                                          2     0     0   |
+-----------------------------------------------------------------------------+
|   0  MIG 3g.20gb     9     2/2         19.62      No     42     2     0   |
|                                                          3     0     0   |
+-----------------------------------------------------------------------------+
|   0  MIG 4g.20gb     5     1/1         19.62      No     56     2     0   |
|                                                          4     0     0   |
+-----------------------------------------------------------------------------+
|   0  MIG 7g.40gb     0     1/1         39.50      No     98     5     0   |
|                                                          7     1     1   |
+-----------------------------------------------------------------------------+
```

The preceding figure shows the following information:

- The ID of the specified A100 GPU is 0. The A100 GPU supports the following GPU partition sizes: `1g.5gb`, `2g.10gb`, `3g.20gb`, `4g.20gb`, and `7g.40gb`. Each GPU partition corresponds to a different memory size. The total memory of each A100 GPU is not adjustable but the partition sizes are different. Therefore, the number of GPU instances that can be created varies based on the partition size that you choose.
- The IDs in the second column indicate the partition sizes.

3. Run the following command to configure MIG for the A100 GPU whose ID is 0:

```
nvidia-smi mig -i 0 -cgi 9,14,19,19
```

Expected output:

```
Successfully created GPU instance ID  2 on GPU  0 using profile MIG 3g.20gb (ID  9)
Successfully created GPU instance ID  3 on GPU  0 using profile MIG 2g.10gb (ID 14)
Successfully created GPU instance ID  9 on GPU  0 using profile MIG 1g.5gb (ID 19)
Successfully created GPU instance ID 10 on GPU  0 using profile MIG 1g.5gb (ID 19)
```

```
nvidia-smi mig -i 0 -cci
```

Expected output:

```
Successfully created compute instance ID  0 on GPU  0 GPU instance ID  9 using profile MIG 1g.5gb (ID  0)
Successfully created compute instance ID  0 on GPU  0 GPU instance ID 10 using profile MIG 1g.5gb (ID  0)
Successfully created compute instance ID  0 on GPU  0 GPU instance ID  3 using profile MIG 2g.10gb (ID  1)
Successfully created compute instance ID  0 on GPU  0 GPU instance ID  2 using profile MIG 3g.20gb (ID  2)
```

The preceding output indicates that the A100 GPU is partitioned into four GPU instances: one `3g.20gb` instance whose ID is 9, one `2g.10gb` instance whose ID is 14, and two `1g.5gb` instances whose IDs are both 19.

4. Run the following command to view information about the GPU instances:

```
nvidia-smi
```

In the `MIG device` section, the following information about the GPU instances is displayed:

```
+-----------------------------------------------------------------------------+
| MIG devices:                                                                |
+------------------+----------------------+-----------+----------------------+
| GPU  GI  CI  MIG |     Memory-Usage     |     Vol   |      Shared          |
|      ID  ID  Dev |      BAR1-Usage      | SM    Unc | CE  ENC  DEC  OFA  JPG|
|                  |                      |     ECC   |                      |
|==================+======================+===========+======================|
| 0    2   0   0   |  11MiB / 20096MiB    | 42    0   | 3   0    2    0    0|
|                  |   0MiB / 32767MiB    |           |                      |
+------------------+----------------------+-----------+----------------------+
| 0    3   0   1   |   7MiB /  9984MiB    | 28    0   | 2   0    1    0    0|
|                  |   0MiB / 16383MiB    |           |                      |
+------------------+----------------------+-----------+----------------------+
| 0    9   0   2   |   3MiB /  4864MiB    | 14    0   | 1   0    0    0    0|
|                  |   0MiB /  8191MiB    |           |                      |
+------------------+----------------------+-----------+----------------------+
| 0   10   0   3   |   3MiB /  4864MiB    | 14    0   | 1   0    0    0    0|
|                  |   0MiB /  8191MiB    |           |                      |
+------------------+----------------------+-----------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

You can use the preceding method to configure MIG for all of the A100 GPUs on the GPU-accelerated ECS instance.

Step 3: Update the device plug-in

1. After you configure MIG for all of the A100 GPUs on the node, run the following command to update the device plug-in and enable MIG:

   ```
   sed -i 's/"--pass-device-specs"/"--pass-device-specs", "--mig-strategy=mixed"/g' /etc/kubernetes/manifests/nvidia-device-plugin.yml
   ```

2. Run the following command to check whether the device plug-in runs as normal:

   ```
   kubectl -n kube-system get po | grep nvidia-device
   ```

   Expected output:

   ```
   nvidia-device-plugin-cn-hangzhou.xxxxxx  1/1   Running  0      36m
   ```

3. After you verify that the device plug-in runs as normal, run the following command to view the number of GPU instances and relevant resources:

   ```
   kubectl describe node <your node>
   ```

   Expected output:

```
Capacity:
 cpu:                104
 ephemeral-storage:  123722704Ki
 hugepages-1Gi:      0
 hugepages-2Mi:      0
 memory:             791733364Ki
 nvidia.com/gpu:     0
 nvidia.com/mig-1g.5gb:  16
 nvidia.com/mig-2g.10gb: 8
 nvidia.com/mig-3g.20gb: 8
```

Step 4: Deploy an application

Use the following YAML file to deploy an application that requests the **mig-2g.10gb** GPU instance:

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Pod
metadata:
 name: smi
spec:
 restartPolicy: OnFailure
 containers:
 - name: nvidia-smi
  command:
  - nvidia-smi
  - -L
  image: nvidia/cuda:9.0-base
  resources:
   limits:
    nvidia.com/mig-2g.10gb: 1
   requests:
    nvidia.com/mig-2g.10gb: 1
EOF
```

Run the following command to print the application log:

```
kubectl logs smi
```

Expected output:

```
GPU 0: A100-SXM4-40GB (UUID: GPU-7780f282-99a1-7024-f7a4-65a55230****)
 MIG 2g.10gb Device 0: (UUID: MIG-GPU-7780f282-99a1-7024-f7a4-65a55230****/3/0)
```

16.7. FAQ about GPUs and NPUs

- How do I upgrade the kernel of a GPU node?
- How do I fix a container startup exception on a GPU node?
- Troubleshoot issues in GPU monitoring
- The number of available GPUs is less than the actual number of GPUs
- Errors in GPU nodes when kubelet or Docker restarts
- Fix the issue that the IDs of GPUs are changed after a GPU-accelerated ECS instance is restarted or replaced

17.Scheduling

17.1. Resource scheduling

17.1.1. Topology-aware CPU scheduling

Container Service for Kubernetes (ACK) provides the topology-aware CPU scheduling feature based on the new Kubernetes scheduling framework. This feature can improve the performance of CPU-sensitive workloads. This topic describes how to enable topology-aware CPU scheduling.

Prerequisites

- A professional managed Kubernetes cluster is created. For more information, see Create a professional managed Kubernetes cluster.

  > **Notice** Topology-aware CPU scheduling is available for only professional managed Kubernetes clusters. To enable topology-aware CPU scheduling for dedicated Kubernetes clusters, Submit a ticket to add your account to the whitelist.

- Before you enable topology-aware CPU scheduling, you must deploy resource-controller. For more information, see Manage system components.
- The following table describes the system component versions that are required for topology-aware CPU scheduling.

Component	Required version
Kubernetes	V1.16 and later
Helm	V3.0 and later
Docker	19.03.5
Operating system	CentOS 7.6, CentOS 7.7, Ubuntu 16.04, Ubuntu 18.04, and Alibaba Cloud Linux 2.

Context

Multiple pods may run on a node in a Kubernetes cluster and some pods may belong to CPU-intensive workloads. In this case, pods compete for CPU resources. When this situation becomes intensive, the CPU cores that are allocated to each pod may be frequently changed. The situation intensifies when the Non-Uniform Memory Access (NUMA) nodes are used. These changes degrade the performance of the workloads. The Kubernetes CPU manager provides a CPU scheduling solution to fix this issue within a node. However, the Kubernetes CPU manager cannot find an optimal solution for allocating CPU cores within a cluster. The Kubernetes CPU manager works only on guaranteed pods and does not apply to all types of pods. In a guaranteed pod, each container is configured with requests and limits on CPU resources. In addition, the requests and limits are set to the same value.

Topology-aware CPU scheduling applies to the following scenarios:

- The workload is compute-intensive.
- The application is CPU-sensitive.
- The workload is deployed on multi-core nodes, such as Elastic Compute Service (ECS) Bare Metal instances.

Enable topology-aware CPU scheduling

Before you enable topology-aware CPU scheduling, you must set the annotations and containers parameters when you configure pods. Set the parameters in the following ways:

- annotation: Set cpuset-scheduler to true to enable topology-aware CPU scheduling.
- containers: Set resources.limit.cpu to an integer.

The following code block shows how to enable topology-aware CPU scheduling for pods:

```
apiVersion: v1
kind: Pod
metadata:
 name: cal-pi
 annotations:
  cpuset-scheduler: 'true' #Add this annotation to enable topology-aware CPU scheduling.
spec:
 restartPolicy: Never
 containers:
 - image: registry.cn-zhangjiakou.aliyuncs.com/xianlu/java-pi
  name: cal-pi
  resources:
   requests:
    cpu: 4
   limits:
    cpu: 4  #Specify the value of resources.limit.cpu.
  env:
  - name: limit
   value: "20000"
  - name: threadNum
   value: "3000"
```

The following code block shows how to enable topology-aware CPU scheduling for Deployments:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: go-demo
spec:
 replicas: 4
 selector:
  matchLabels:
   app: go-demo
 template:
  metadata:
   annotations:
    cpuset-scheduler: "true" #Add this annotation to enable topology-aware CPU scheduling.
   labels:
    app: go-demo
  spec:
   containers:
   - name: go-demo
    image: registry.cn-hangzhou.aliyuncs.com/haoshuwei24/go-demo:1k
    imagePullPolicy: Always
    ports:
    - containerPort: 8080
    resources:
     requests:
      cpu: 1
     limits:
      cpu: 4  #Specify the value of resources.limit.cpu.
```

17.1.2. Use resource-controller to dynamically modify the upper limit of resources for a pod

The resource-controller component is a controller that uses Kubernetes Custom Resource Definitions (CRDs) to control resource usage. It allows you to modify the upper limit of resources for a pod without restarting the pod. For example, you can modify the upper limit of CPU and memory resources. This enables containers in a pod to handle workloads as normal with the specified amount of resources. This topic describes how to dynamically modify the upper limit of resources for a pod after you deploy the resource-controller component.

Prerequisites

- Deploy the resource-controller component. For more information, see Manage system components.
- Use kubectl to connect to a Container Service for Kubernetes (ACK) cluster. For more information, see Connect to Kubernetes clusters by using kubectl.

Context

We recommend that you deploy resource-controller to dynamically modify the upper limit of resources for a pod in the following scenarios:

- The pod is running. However, the specified CPU limit is low, which limits the speed of processes in the pod.
- The CPU loads of the pod are high because you did not specify resource thresholds when you initialized the pod. You want to limit CPU usage of the pod without affecting other applications.
- The memory usage of the pod is increasing and reaching the specified upper limit. You want to raise the memory upper limit without restarting the pod before the Out of Memory (OOM) killer is triggered.

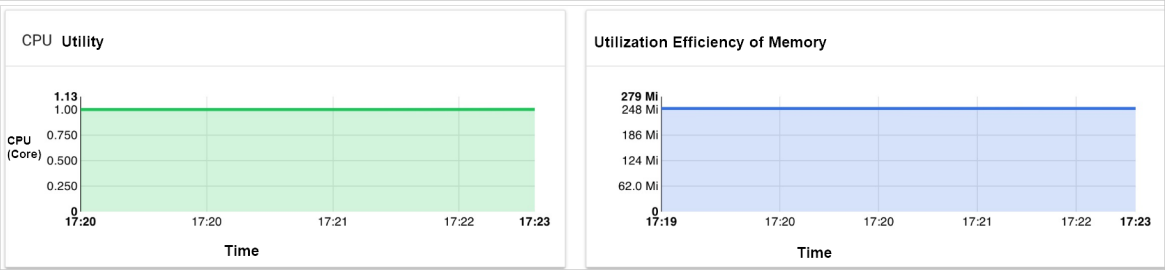
In all preceding scenarios, the design principles of Kubernetes allow you to change the upper limit of resources for a pod only by modifying PodSpec. The pod is recreated after you change the upper limit. If an online application runs in the pod, users may fail to access the application and the network traffic may spike after the pod is recreated. If an offline task runs in the pod, all computing data generated within the previous hours may be lost after the pod is recreated.

Dynamically modify the CPU and memory usage for a pod

1. Deploy a task in a pod for simulation. The task is a stress testing program that uses 2 CPU cores and 256 MB of memory.

   Use the following template to deploy the simulation task and set the upper limit of CPU usage to 1:

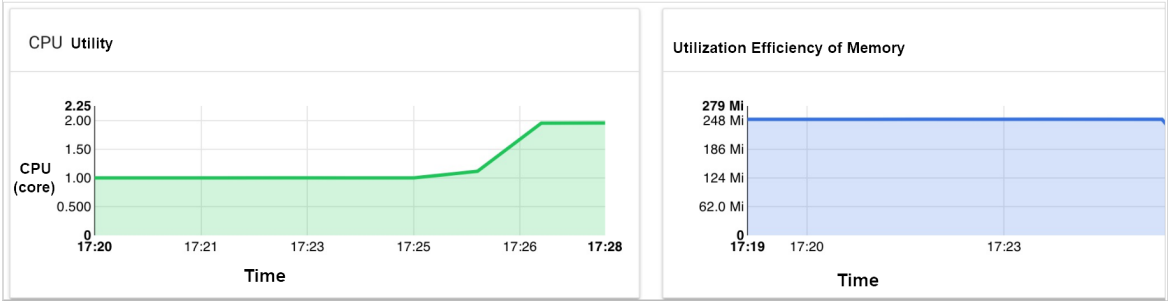
   ```
   apiVersion: v1
   kind: Pod
   metadata:
    name: pod-demo
   spec:
    containers:
    - name: pod-demo
     image: polinux/stress
     resources:
      requests:
       memory: "50Mi"
      limits:
       memory: "1000Mi"
       cpu: 1
     command: ["stress"]
     args: ["--vm", "1", "--vm-bytes", "256M", "-c", "2", "--vm-hang", "1"]
   ```

   The preceding figure shows that the pod can use only one CPU core.

2. Submit the following CRD template to dynamically modify the upper limit of CPU and memory usage:

   ```
   apiVersion: resources.alibabacloud.com/v1alpha1
   kind: Cgroups
   metadata:
    name: cgroups-sample
   spec:
    pods:
    - name: pod-demo
     containers:
     - name: pod-demo
       cpu: 2000m
       memory: 5000Mi
   ```



The preceding figure shows that the CPU usage of the pod increases from one CPU core to two CPU cores.

3. Run the following command to check the state of the pod:

```
kubectl describe pod pod-demo
```

The following is an example of the output, which indicates that the pod is running as expected and is not restarted:

```
Events:
  Type    Reason          Age   From                    Message
  ----    ------          ----  ----                    -------
  Normal  Scheduled       13m   default-scheduler       Successfully assigned default/pod-demo to cn-zhan
gjiakou.192.168.3.238
  Normal  Pulling         13m   kubelet, cn-zhangjiakou.192.168.3.238  Pulling image "polinux/stress"
  Normal  Pulled          13m   kubelet, cn-zhangjiakou.192.168.3.238  Successfully pulled image "polinux/stress"
  Normal  SuccessfulChange  60s   cgroups-controller      Change pod pod-demo cpu to 2000
  Normal  SuccessfulChange  60s   cgroups-controller      Change pod pod-demo memory to 52428800000
0
```

Bind a pod to one or more CPUs

1. Use the following template to run a stress testing program in a pod. The program uses 0.5 CPU cores.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-demo
  annotations:
    cpuset-scheduler: 'true' #Add this annotation to enable topology-aware CPU scheduling.
spec:
  containers:
  - name: pod-demo
    image: polinux/stress
    resources:
      requests:
        memory: "50Mi"
      limits:
        memory: "1000Mi"
        cpu: 0.5
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes", "556M", "-c", "2", "--vm-hang", "1"]
```

2. View the CPU usage of the node `cn-beijing.192.168.8.241`. The following result indicates that the CPUs show different usage and the usage changes dynamically:

```
top - 22:17:34 up 4 days, 10:29,  1 user,  load average: 0.33, 0.88, 0.95
Tasks: 179 total,   3 running, 176 sleeping,   0 stopped,   0 zombie
%Cpu0  : 13.1 us,  0.7 sy,  0.0 ni, 85.9 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :  7.3 us,  7.7 sy,  0.0 ni, 84.7 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  : 12.4 us,  0.7 sy,  0.0 ni, 86.6 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  : 18.3 us,  0.7 sy,  0.0 ni, 80.7 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
```

3. Submit the following CRD template to bind the pod to CPU2 and CPU3:

```
apiVersion: resources.alibabacloud.com/v1alpha1
kind: Cgroups
metadata:
 name: cgroups-sample-cpusetpod
spec:
 pod:
  name: pod-demo
  namespace: default
  containers:
  - name: pod-demo
   cpuset-cpus: 2-3
```

4. View the CPU usage of the node. The following result shows that the sum of the usage of CPU2 and CPU3 stays around 50% and the usage of each CPU stays around 25%. This indicates that the pod has been bound to CPU2 and CPU3 as expected and the pod is not restarted.

```
top - 22:11:02 up 4 days, 10:22,  1 user,  load average: 0.04, 0.36, 0.84
Tasks: 177 total,   3 running, 174 sleeping,   0 stopped,   0 zombie
%Cpu0  :  2.7 us,  0.7 sy,  0.0 ni, 96.3 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :  3.3 us,  1.0 sy,  0.0 ni, 95.3 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  : 27.2 us,  0.7 sy,  0.0 ni, 71.8 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  : 21.4 us,  5.7 sy,  0.0 ni, 72.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Bind a Deployment to one or more CPUs

1. The following template shows how to use a Deployment to run a stress testing program with two instances. Each instance uses 0.5 CPU cores.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: go-demo
 labels:
  app: go-demo
spec:
 replicas: 2
 selector:
  matchLabels:
   app: go-demo
 template:
  metadata:
   annotations:
    cpuset-scheduler: "true" #Add this annotation to enable topology-aware CPU scheduling.
   labels:
    app: go-demo
  spec:
   nodeName: cn-beijing.192.168.8.240 #Schedule to the same node.
   containers:
   - name: go-demo
    image: polinux/stress
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes", "556M", "-c", "1", "--vm-hang", "1"]
    imagePullPolicy: Always
    resources:
     requests:
      cpu: 0.5
     limits:
      cpu: 0.5  #Specify the value of resources.limit.cpu.
```

2. View the CPU usage of the node `cn-beijing.192.168.8.240`. The following result indicates that the CPUs show different usage and the usage changes dynamically:

```
top - 11:39:01 up 23:50,  2 users,  load average: 1.76, 1.91, 1.39
Tasks: 189 total,   4 running, 185 sleeping,   0 stopped,   0 zombie
%Cpu0  : 30.4 us,  5.4 sy,  0.0 ni, 64.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  : 29.4 us,  4.7 sy,  0.0 ni, 65.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  :  7.0 us,  8.7 sy,  0.0 ni, 84.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  : 50.3 us,  1.3 sy,  0.0 ni, 48.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

3. Submit the following CRD template to bind the Deployment to CPU2 and CPU3:

```
apiVersion: resources.alibabacloud.com/v1alpha1
kind: Cgroups
metadata:
 name: cgroups-cpuset-sample
spec:
 deployment:
  name: go-demo
  namespace: default
  containers:
  - name: go-demo
   cpuset-cpus: 2,3 #Bind the pod to CPU2 and CPU3.
```

4. View the CPU usage of the node. The following result shows that the sum of the usage of CPU2 and CPU3 stays around 50%. This indicates that the two instances in the Deployment are separately bound

to CPU2 and CPU3 as expected, and the pod is not restarted.

```
top - 11:30:56 up 23:42,  2 users,  load average: 2.01, 1.95, 1.12
Tasks: 180 total,   4 running, 176 sleeping,   0 stopped,   0 zombie
%Cpu0  :  4.4 us,  2.4 sy,  0.0 ni, 93.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :  4.4 us,  2.3 sy,  0.0 ni, 92.6 id,  0.3 wa,  0.0 hi,  0.3 si,  0.0 st
%Cpu2  : 52.7 us,  8.0 sy,  0.0 ni, 39.0 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  : 50.7 us, 10.7 sy,  0.0 ni, 38.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Control the IOPS of a disk

Note To control the input/output operations per second (IOPS) of a disk, you must create worker nodes that use the Alibaba Cloud Linux 2 operating system.

1. Deploy a Fio container. The container uses Fio to perform write stress tests on the disk.

   Use the following template to create a disk volume and mount the disk volume to the node `cn-beijing.192.168.0.182`. The device number of the disk is `/dev/vdb1` and the disk is mounted to the path */mnt*.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: fio
 labels:
  app: fio
spec:
 selector:
  matchLabels:
   app: fio
 template:
  metadata:
   labels:
    app: fio
  spec:
   nodeName: cn-beijing.192.168.0.182  # The pod is deployed on the node cn-beijing.192.168.0.182.
   containers:
   - name: fio
    image: registry.cn-beijing.aliyuncs.com/shuangkun/fio:v1
    command: ["sh", "-c"]
    # Use Fio to perform write stress tests on the disk.
    args: ["fio -filename=/data/test -direct=1 -iodepth 1 -thread -rw=write -ioengine=psync -bs=16k -size=2G -
numjobs=10 -runtime=12000 -group_reporting -name=mytest"]
    volumeMounts:
     - name: pvc
      mountPath: /data   #The disk volume is mounted to the path /data.
   volumes:
    - name: pvc
     hostPath:
      path: /mnt
```

2. Limit the throughput of the pod by deploying the CRD that is used to control the IOPS of the disk.

   Use the following template to set the write `BPS`, read BPS, and total BPS of the `/dev/vdb1` disk to 1048576, 2097152, and 3145728:

```
apiVersion: resources.alibabacloud.com/v1alpha1
kind: Cgroups
metadata:
 name: cgroups-sample-fio
spec:
 pod:
  name: fio-6b6c469fdf-44h7v
  namespace: default
  containers:
  - name: fio
   blkio:
    device_write_bps: [{device: "/dev/vdb1", value: "1048576"}]
```

3. View the monitoring data of the disk on the node `cn-beijing.192.168.0.182`, as shown in the following figure.

The preceding figure shows that the throughput `BPS` of the pod is as expected and the pod is not restarted during modification.

Topology-aware CPU scheduling

You can use resource-controller with ACK schedulers to facilitate CPU binding and automate CPU selection on multi-core physical machines, such as Intel, AMD, and ARM. For more information, see Topology-aware CPU scheduling.

Topology-aware CPU scheduling manages CPU cores and hyper-threads in an appropriate manner to avoid the following issues: switchover between L1 cache and L2 cache, off-chip transmission across Non-Uniform Memory Access (NUMA), and frequent refreshing of L3 cache. This maximizes the CPU usage efficiency for CPU-intensive applications that run with multiple threads. For more information about topology-aware CPU scheduling, see the speech Practice of Fine-grained Cgroups Resources Scheduling in Kubernetes that is given by the ACK team at KubeCon 2020.

Related information

- Topology-aware CPU scheduling

17.2. GPU scheduling

This topic describes the various methods of GPU scheduling, such as the default GPU scheduling, GPU sharing and scheduling, and topology-aware GPU scheduling. GPU sharing and scheduling improve the utilization of GPU resources. Topology-aware GPU scheduling accelerates task processing.

Default GPU scheduling

After you create a Kubernetes cluster with GPU-accelerated nodes, you can set up a GPU-accelerated environment to run TensorFlow. For more information about how to schedule dedicated GPU resources, see Use GPU scheduling for ACK clusters.

You can also enable custom GPU scheduling based on node labels. For more information, see Use labels to schedule pods to GPU-accelerated nodes.

GPU sharing and scheduling

Container Service for Kubernetes (ACK) provides the open source cGPU solution that allows you to share one GPU among multiple containers in an ACK cluster. You can enable cGPU for container clusters that are deployed in Alibaba Cloud, Amazon Web Services (AWS), Google Compute Engine (GCE), or data centers. cGPU enables GPU sharing and reduces the cost of GPU resources. cGPU also enables the isolation of GPU resources allocated to multiple containers when one GPU is shared. This prevents the issue in which some containers consume excessive resources and other containers run with insufficient resources. cGPU also provides fine-grained GPU utilization. You can refer to the following topics for further details:

For more information about cGPU, see Overview.

For more information about how to enable cGPU, see Install the cGPU component. For more information about how to disable cGPU, see Disable the memory isolation capability of cGPU.

For more information about how to use cGPU, see Enable GPU sharing, Monitor and isolate GPU resources, and 基于节点池管理共享GPU.

Topology-aware GPU scheduling

Kubernetes is unaware of the topology of GPU resources on nodes. Therefore, Kubernetes schedules GPU resources in a random manner. As a result, the GPU acceleration for training jobs considerably varies based on the scheduling results of GPU resources. To avoid this situation, ACK supports topology-aware GPU scheduling based on the scheduling framework of Kubernetes. You can use this feature to select a combination of GPUs from GPU-accelerated nodes to achieve optimal GPU acceleration for training jobs. For more information about how to use topology-aware GPU scheduling, see the following topics:

- Overview
- Install the ack-ai-installer component
-
-

17.3. FPGA scheduling

17.3.1. Use labels to schedule pods to FPGA-accelerated nodes

When FPGAs are used for computing in a Container Service for Kubernetes (ACK) cluster, you can schedule pods to FPGA-accelerated nodes. This allows you to make full use of FPGA resources. This topic describes how to schedule pods to FPGA-accelerated nodes by using labels.

Prerequisites

- An ACK cluster with FPGA-accelerated nodes is created. For more information, see Create a managed Kubernetes cluster with FPGA-acceleated nodes.
- You are connected to the ACK cluster. This way, you can view the labels that are added to the nodes. For more information, see Connect to Kubernetes clusters by using kubectl.

Context

When you deploy FPGA-accelerated nodes in an ACK cluster, FPGA-related attributes are exposed by node labels. This offers the following benefits:

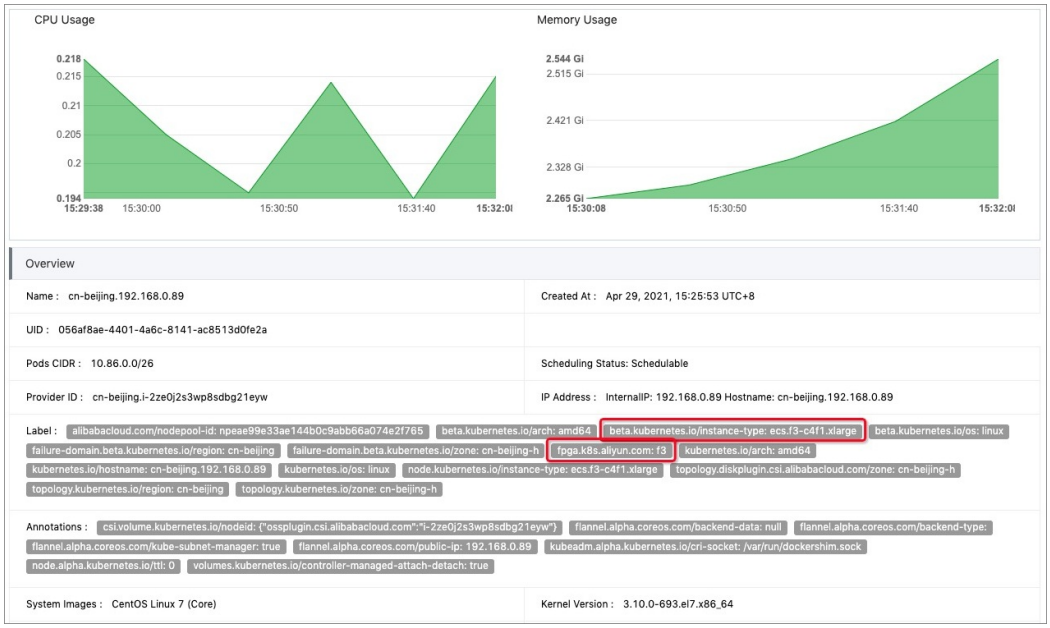
- You can use the labels to filter FPGA-accelerated nodes.
- The labels can be used as conditions to schedule pods.

Step 1: View the labels of the FPGA-accelerated nodes

Method 1: View the labels of the FPGA-accelerated nodes in the console

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the **Nodes** page, find the FPGA-accelerated node that you want to manage, and choose **More > Details** in the **Actions** column.

   View the labels of the FPGA-accelerated node.



Method 2: Run the kubectl command to view the labels of the FPGA-accelerated nodes

1. Run the following command to view the details of the FPGA-accelerated nodes:

   ```
   kubectl get nodes
   ```

   Expected output:

```
NAME            STATUS  ROLES   AGE   VERSION
cn-beijing.192.168.XX.X1  Ready   <none>  3h51m  v1.18.8-aliyun.1
cn-beijing.192.168.XX.X2  Ready   <none>  3h41m  v1.18.8-aliyun.1
```

2. Select an FPGA-accelerated node and run the following command to view the labels of the FPGA-accelerated node:

```
kubectl describe node cn-beijing.192.168.XX.X2
```

Expected output:

```
Name:         cn-beijing.192.168.XX.X2
Roles:        <none>
Labels:       ack.aliyun.com=c05888610e***
              alibabacloud.com/nodepool-id=npfda879b6***
              beta.kubernetes.io/arch=amd64
              beta.kubernetes.io/instance-type=ecs.f3-c4f1.xlarge
              beta.kubernetes.io/os=linux
              failure-domain.beta.kubernetes.io/region=cn-beijing
              failure-domain.beta.kubernetes.io/zone=cn-beijing-h
              fpga.k8s.aliyun.com=f3
              kubernetes.io/arch=amd64
              kubernetes.io/hostname=cn-beijing.192.168.XX.X2
              kubernetes.io/os=linux
              node.kubernetes.io/instance-type=ecs.f3-c4f1.xlarge
              topology.diskplugin.csi.alibabacloud.com/zone=cn-beijing-h
              topology.kubernetes.io/region=cn-beijing
              topology.kubernetes.io/zone=cn-beijing-h
```

The output shows that the `fpga.k8s.aliyun.com=f3` label is added to the FPGA-accelerated node. The label is used to schedule pods in the following example.

Step 2: Schedule pods to the FPGA-accelerated nodes

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Select a custom template from the drop-down list of sample templates, and copy the following content to the **Template** field.

```
apiVersion: batch/v1
kind: Job
metadata:
 name: fpga-run-task1
spec:
 backoffLimit: 0
 completions: 1
 parallelism: 1
 template:
  spec:
   nodeSelector:
    fpga.k8s.aliyun.com: f3
   containers:
   - image: <your image>
    imagePullPolicy: Always
    name: fpga-run-task1
    resources:
     limits:
      xilinx.com/fpga-aliyun-f3: 1
    securityContext:
     privileged: true
```

Note Replace the value of `<your image>` with the address of your custom image.

7. Click **Create**.
   In the left-side navigation pane of the details page, choose **Workloads > Pods**.
   In the list of pods, you can view that the specified pod is scheduled to the required FPGA-accelerated node. You can use labels to schedule pods to specific FPGA-accelerated nodes with ease.

17.4. Workload scheduling

17.4.1. Gang scheduling

The gang scheduling feature provided by Container Service for Kubernetes (ACK) is developed on top of the new kube-scheduler framework. This feature provides a solution to job scheduling in all-or-nothing scenarios. This topic describes how to enable gang scheduling.

Prerequisites

- A professional managed Kubernetes cluster is created. For more information, see Create a professional managed Kubernetes cluster.

  Notice Gang scheduling is available for only professional managed Kubernetes clusters. To enable gang scheduling for dedicated Kubernetes clusters, Submit a ticket to apply for this feature to be enabled on your account.

- The following table describes the system component versions that are required to enable gang scheduling.

Component	Required version
Kubernetes	V1.16 and later
Helm	V3.0 and later

Component	Required version
Docker	19.03.5
Operating system	CentOS 7.6, CentOS 7.7, Ubuntu 16.04, Ubuntu 18.04, and Alibaba Cloud Linux 2.

Context

Gang scheduling is a scheduling algorithm that schedules multiple correlated processes to different processors in a parallel system and simultaneously starts these processes. Gang scheduling aims to start all correlated processes at the same time. This ensures that the process group is not blocked when the system fails to start some processes. For example, if you submit a batch job that contains multiple tasks, either all of the tasks are scheduled or none of them is scheduled. Task scheduling in all-or-nothing scenarios is known as gang scheduling.

Kubernetes is widely used in online service orchestration. ACK wants to use Kubernetes as a platform for the unified management of online services and offline jobs. This improves the resource utilization and performance of clusters. However, kube-scheduler cannot migrate specific offline workloads to Kubernetes clusters. For example, if a job requires all-or-nothing scheduling, all tasks of the job must be scheduled at the same time. If only some of the tasks are started, the started jobs must wait until all the remaining tasks are scheduled. If each submitted job contains unscheduled tasks, all submitted jobs remain in the Pending state and the cluster is deadlocked. To avoid this situation, you must enable gang scheduling for kube-scheduler.

Feature description

In ACK, a pod group is a group of pods that need to be scheduled at the same time. When you submit a job that requires all-or-nothing scheduling, you can add labels to pods. The labels specify the name of the pod group to which the job belongs and the minimum number of tasks that must be scheduled to run the job. kube-scheduler schedules tasks based on the minimum number of tasks that must be scheduled. The tasks are scheduled only when the cluster resources are sufficient to schedule the required number of tasks. Otherwise, the job remains in the Pending state.

How to enable gang scheduling

To enable gang scheduling, set min-available and name by adding labels to the pods.

```
labels:
  pod-group.scheduling.sigs.k8s.io/name: tf-smoke-gpu
  pod-group.scheduling.sigs.k8s.io/min-available: "3"
```

- name: the name of a pod group.
- min-available: the minimum number of pods that must be scheduled to run a job. Pods are scheduled only when the computing resources are sufficient to schedule the required number of pods.

> **Note** Pods in the same pod group must be assigned the same priority.

Examples

In this example, a distributed TensorFlow job is used to demonstrate how to enable gang scheduling. The ACK cluster that is used in this example has four GPUs.

1. Run the following command to use Arena of Kubeflow to deploy the environment in your ACK cluster to run the distributed TensorFlow job.

> **Note** Arena is a subproject of Kubeflow. Kubeflow is an open source project for Kubernetes-based machine learning. Arena allows you to manage the lifecycle of machine learning jobs by using a CLI or SDK. Lifecycle management includes environment setup, data preparation, model development, model training, and model prediction. This improves the working efficiency of data scientists.

```
git clone https://github.com/kubeflow/arena.git
kubectl create ns arena-system
kubectl create -f arena/kubernetes-artifacts/jobmon/jobmon-role.yaml
kubectl create -f arena/kubernetes-artifacts/tf-operator/tf-crd.yaml
kubectl create -f arena/kubernetes-artifacts/tf-operator/tf-operator.yaml
```

Run the following command to check whether the environment to run TensorFlow jobs is deployed. If the pods are in the Running state, it indicates that the environment is deployed.

```
kubectl get pods -n arena-system
```

```
NAME                READY  STATUS   RESTARTS  AGE
tf-job-dashboard-56cf48874f-gwlhv  1/1   Running  0     54s
tf-job-operator-66494d88fd-snm9m   1/1   Running  0     54s
```

2. Use the following template to submit a distributed TensorFlow job to the ACK cluster. The job runs on one parameter server (PS) pod and four worker pods. Each worker pod requires two GPUs.

```
apiVersion: "kubeflow.org/v1"
kind: "TFJob"
metadata:
 name: "tf-smoke-gpu"
spec:
 tfReplicaSpecs:
  PS:
   replicas: 1
   template:
    metadata:
     creationTimestamp: null
     labels:
      pod-group.scheduling.sigs.k8s.io/name: tf-smoke-gpu
      pod-group.scheduling.sigs.k8s.io/min-available: "5"
    spec:
     containers:
     - args:
      - python
      - tf_cnn_benchmarks.py
      - --batch_size=32
      - --model=resnet50
      - --variable_update=parameter_server
      - --flush_stdout=true
      - --num_gpus=1
      - --local_parameter_device=cpu
      - --device=cpu
      - --data_format=NHWC
      image: registry.cn-hangzhou.aliyuncs.com/kubeflow-images-public/tf-benchmarks-cpu:v20171202-bda
b599-dirty-284af3
      name: tensorflow
      ports:
      - containerPort: 2222
```

```
        name: tfjob-port
       resources:
        limits:
         cpu: '1'
       workingDir: /opt/tf-benchmarks/scripts/tf_cnn_benchmarks
      restartPolicy: OnFailure
   Worker:
    replicas: 4
    template:
     metadata:
      creationTimestamp: null
      labels:
       pod-group.scheduling.sigs.k8s.io/name: tf-smoke-gpu
       pod-group.scheduling.sigs.k8s.io/min-available: "5"
     spec:
      containers:
      - args:
       - python
       - tf_cnn_benchmarks.py
       - --batch_size=32
       - --model=resnet50
       - --variable_update=parameter_server
       - --flush_stdout=true
       - --num_gpus=1
       - --local_parameter_device=cpu
       - --device=gpu
       - --data_format=NHWC
       image: registry.cn-hangzhou.aliyuncs.com/kubeflow-images-public/tf-benchmarks-gpu:v20171202-bda
b599-dirty-284af3
       name: tensorflow
       ports:
       - containerPort: 2222
         name: tfjob-port
       resources:
        limits:
          nvidia.com/gpu: 2
       workingDir: /opt/tf-benchmarks/scripts/tf_cnn_benchmarks
      restartPolicy: OnFailure
```

- Submit the distributed TensorFlow job without enabling gang scheduling
  Run the following command to query the states of pods. Only two worker pods are running and the other worker pods are in the Pending state.

  ```
  kubectl get pods
  ```

  ```
  NAME          READY  STATUS   RESTARTS  AGE
  tf-smoke-gpu-ps-0    1/1   Running  0     6m43s
  tf-smoke-gpu-worker-0  1/1   Running  0     6m43s
  tf-smoke-gpu-worker-1  1/1   Running  0     6m43s
  tf-smoke-gpu-worker-2  0/1   Pending  0     6m43s
  tf-smoke-gpu-worker-3  0/1   Pending  0     6m43s
  ```

  Run the following command to query the log data of the running worker pods. The returned log data indicates that the running worker pods are waiting for the system to start the pending worker pods. The GPU resources occupied by the running worker pods are not in use.

```
kubectl logs -f tf-smoke-gpu-worker-0
```

```
INFO|2020-05-19T07:02:18|/opt/launcher.py|27| 2020-05-19 07:02:18.199696: I tensorflow/core/distributed_
runtime/master.cc:221] CreateSession still waiting for response from worker: /job:worker/replica:0/task:3
INFO|2020-05-19T07:02:28|/opt/launcher.py|27| 2020-05-19 07:02:28.199798: I tensorflow/core/distributed_
runtime/master.cc:221] CreateSession still waiting for response from worker: /job:worker/replica:0/task:2
```

- Submit the distributed TensorFlow job with gang scheduling enabled
  Run the following command to query the states of pods: The computing resources in the cluster are insufficient to schedule the minimum number of pods. Therefore, the pod group cannot be scheduled and all pods are in the Pending state.

  ```
  kubectl get pods
  ```

  ```
  NAME          READY  STATUS   RESTARTS  AGE
  tf-smoke-gpu-ps-0    0/1   Pending  0     43s
  tf-smoke-gpu-worker-0  0/1   Pending  0     43s
  tf-smoke-gpu-worker-1  0/1   Pending  0     43s
  tf-smoke-gpu-worker-2  0/1   Pending  0     43s
  tf-smoke-gpu-worker-3  0/1   Pending  0     43s
  ```

  After four GPUs are allocated to the cluster, the computing resources in the cluster are sufficient to schedule the minimum number of pods. After the pod group is scheduled, the four worker pods start to run. Run the following command to query the states of pods:

  ```
  kubectl get pods
  ```

  ```
  NAME          READY  STATUS   RESTARTS  AGE
  tf-smoke-gpu-ps-0    1/1   Running  0     3m16s
  tf-smoke-gpu-worker-0  1/1   Running  0     3m16s
  tf-smoke-gpu-worker-1  1/1   Running  0     3m16s
  tf-smoke-gpu-worker-2  1/1   Running  0     3m16s
  tf-smoke-gpu-worker-3  1/1   Running  0     3m16s
  ```

  Run the following command to query the log data of a running worker pod. The following output indicates that the tasks have been started.

  ```
  kubectl logs -f tf-smoke-gpu-worker-0
  ```

  ```
  INFO|2020-05-19T07:15:24|/opt/launcher.py|27| Running warm up
  INFO|2020-05-19T07:21:04|/opt/launcher.py|27| Done warm up
  INFO|2020-05-19T07:21:04|/opt/launcher.py|27| Step  Img/sec loss
  INFO|2020-05-19T07:21:05|/opt/launcher.py|27| 1 images/sec: 31.6 +/- 0.0 (jitter = 0.0) 8.318
  INFO|2020-05-19T07:21:15|/opt/launcher.py|27| 10  images/sec: 31.1 +/- 0.4 (jitter = 0.7) 8.343
  INFO|2020-05-19T07:21:25|/opt/launcher.py|27| 20  images/sec: 31.5 +/- 0.3 (jitter = 0.7) 8.142
  ```

17.4.2. Capacity Scheduling

Kubernetes uses resource quotas to allocate resources based on fixed amounts. This method does not ensure high resource utilization in a Kubernetes cluster. To improve the resource utilization of a Kubernetes cluster, Alibaba Cloud has developed the capacity scheduling feature to optimize resource allocation. This feature is designed on top of the Yarn capacity scheduler and the Kubernetes scheduling framework. This feature allows you to meet the resource requests in a Kubernetes cluster and improve resource utilization by sharing idle resources. This topic describes how to use the capacity scheduling feature.

Prerequisites

- Create a professional managed Kubernetes cluster.
- The Kubernetes version of the created cluster is V1.20 or later.

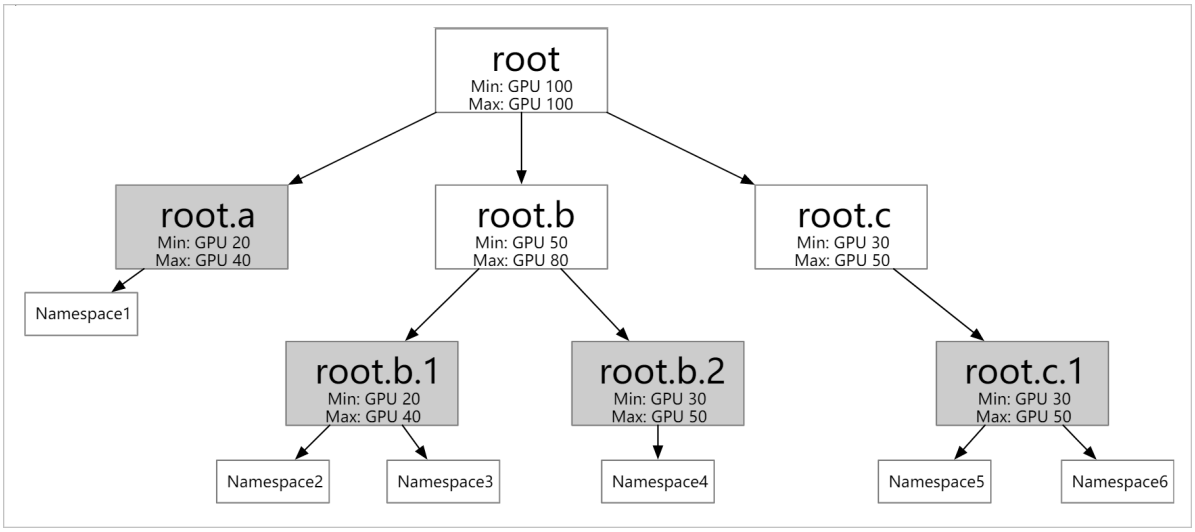
Context

If a Kubernetes cluster is used by multiple users, the cluster administrator allocates a fixed amount of resources to each user to meet the resource requirements. The traditional method is to use Kubernetes resource quotas to allocate resources. These users may use resources in different ways at different times. As a result, some users may encounter resource shortages, but the resources allocated to other users are not being used. In this case, the resource utilization of the cluster becomes lower and a considerable amount of resources are wasted.

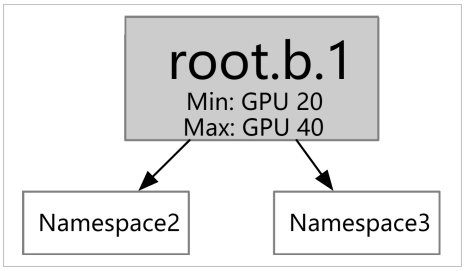
Core features

To address this issue, Alibaba Cloud provides the capacity scheduling feature based on the Kubernetes scheduling framework to optimize resource allocation. This feature allows you to meet the resource requests in a Kubernetes cluster and improve resource utilization by sharing idle resources. The capacity scheduling feature provides the following features:

- Support hierarchical resource quotas. You can configure hierarchical resource quotas based on your requirements (such as enterprise scenarios), as shown in the following figure. In an elastic resource quota group, each namespace belongs to only one leaf and a leaf can contain multiple namespaces.



- Support resource sharing and reclaiming between resource quotas.
  - Min: the minimum amount of resources that are guaranteed for use. If the resources of a cluster become insufficient, the total amount of minimum resources for all users must be lower than the total amount of resources of the cluster.
  - Max: The maximum amount of resources that you can use.



> **Note**
>
> - Idle resource quotas of other users can be temporarily used by your workloads. However, the total amount of resources used by your workloads cannot exceed the maximum amount of the corresponding resource quota.
> - If the minimum amount of resources allocated to your workloads are idle, they can be temporarily used by other users. When you require these resources, they are reclaimed and preempted by your workloads.

- Support multiple resource types. You can configure CPU and memory resource quotas. You can also configure resource quotas for extended resources that are supported by Kubernetes, such as GPUs.

Examples of capacity scheduling

In this topic, an Elastic Compute Service (ECS) instance of the ecs.sn2.13xlarge type (56 vCPUs and 224 GiB) is used to show how to configure resource quotas.

1. Run the following command to create namespaces:

```
kubectl create ns namespace1
kubectl create ns namespace2
kubectl create ns namespace3
kubectl create ns namespace4
```

2. Create an elastic quota group by using the following YAML template:

```
apiVersion: scheduling.sigs.k8s.io/v1beta1
kind: ElasticQuotaTree
metadata:
 name: elasticquotatree
 namespace: kube-system # The elastic quota group takes effect only if it is created in the kube-system name
space.
spec:
 root:
  name: root # Configure the resource quota of the root. The maximum amount of resources for the root mus
t equal the minimum amount of resources for the root.
  max:
   cpu: 40
   memory: 40Gi
   nvidia.com/gpu: 4
  min:
   cpu: 40
   memory: 40Gi
   nvidia.com/gpu: 4
  children: # Configure resource quotas for the leaves of the root.
   - name: root.a
     max:
      cpu: 40
      memory: 40Gi
      nvidia.com/gpu: 4
     min:
      cpu: 20
      memory: 20Gi
      nvidia.com/gpu: 2
     children: # Configure resource quotas of the farthest leaves.
      - name: root.a.1
        namespaces: # Configure resource quotas of the namespaces.
          - namespace1
```

```
      - namespace1
     max:
      cpu: 20
      memory: 20Gi
      nvidia.com/gpu: 2
     min:
      cpu: 10
      memory: 10Gi
      nvidia.com/gpu: 1
    - name: root.a.2
     namespaces: # Configure resource quotas of the namespaces.
      - namespace2
     max:
      cpu: 20
      memory: 40Gi
      nvidia.com/gpu: 2
     min:
      cpu: 10
      memory: 10Gi
      nvidia.com/gpu: 1
  - name: root.b
   max:
    cpu: 40
    memory: 40Gi
    nvidia.com/gpu: 4
   min:
    cpu: 20
    memory: 20Gi
    nvidia.com/gpu: 2
   children: # Configure resource quotas of the farthest leaves.
    - name: root.b.1
     namespaces: # Configure resource quotas of the namespaces.
      - namespace3
     max:
      cpu: 20
      memory: 20Gi
      nvidia.com/gpu: 2
     min:
      cpu: 10
      memory: 10Gi
      nvidia.com/gpu: 1
    - name: root.b.2
     namespaces: # Configure resource quotas of the namespaces.
      - namespace4
     max:
      cpu: 20
      memory: 20Gi
      nvidia.com/gpu: 2
     min:
      cpu: 10
      memory: 10Gi
      nvidia.com/gpu: 1
```

In the preceding YAML template, namespaces created from the node are specified in the `namespaces` fields. The resource quotas of the leaves under a parent leaf are specified in the `children` parameters. The quota configurations must meet the following requirements:

- The minimum amount of resources for a leaf cannot exceed the maximum amount of resources for the leaf.
- The total amount of minimum resources for all leaves under a parent leaf cannot exceed the minimum amount of resources for the parent leaf.
- The minimum amount of resources for the root must equal the maximum amount of resources for the root. This amount cannot exceed the total amount of resources of the cluster.
- Each namespace belongs to only one leaf. A leaf can contain multiple namespaces.

3. Run the following command to check whether the elastic quota group is created:

```
kubectl get ElasticQuotaTree -n kube-system
```

Expected output:

```
NAME          AGE
elasticquotatree  68s
```

4. Create a Deployment in `namespace1` by using the following YAML template. The Deployment runs five pods and each pod requests five CPU cores.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx1
 namespace: namespace1
 labels:
  app: nginx1
spec:
 replicas: 5
 selector:
  matchLabels:
   app: nginx1
 template:
  metadata:
   name: nginx1
   labels:
    app: nginx1
  spec:
   containers:
   - name: nginx1
    image: nginx
    resources:
     limits:
      cpu: 5
     requests:
      cpu: 5
```

5. Run the following command to query the states of these pods:

```
kubectl get pods -n namespace1
```

Expected output:

```
NAME          READY  STATUS   RESTARTS  AGE
nginx1-744b889544-52dbg  1/1   Running  0      70s
nginx1-744b889544-6l4s9  1/1   Running  0      70s
nginx1-744b889544-cgzlr  1/1   Running  0      70s
nginx1-744b889544-w2gr7  1/1   Running  0      70s
nginx1-744b889544-zr5xz  0/1   Pending  0      70s
```

- The total CPU resource requests of the pods in `namespace1` exceed `10 CPU cores`, which is the minimum number of CPU cores for `root.a.1`. The maximum number of CPU cores for the root is `40`. Therefore, these pods can use idle CPU cores in the cluster. The maximum number of CPU cores used by these pods cannot exceed `20`, which is the maximum number of CPU cores for `root.a.1`.
- If the number of CPU cores used by the pods in namespace1 reaches `20`, the next pod to be scheduled becomes pending. Among the five pods of this Deployment, four are in the running state and one is in the pending state.

6. Create a second Deployment in `namespace2` by using the following YAML template. The Deployment runs five pods and each pod requests five CPU cores.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx2
 namespace: namespace2
 labels:
  app: nginx2
spec:
 replicas: 5
 selector:
  matchLabels:
   app: nginx2
 template:
  metadata:
   name: nginx2
   labels:
    app: nginx2
  spec:
   containers:
   - name: nginx2
    image: nginx
    resources:
     limits:
      cpu: 5
     requests:
      cpu: 5
```

7. Run the following command to query the states of these pods:

```
kubectl get pods -n namespace1
```

Expected output:

```
NAME          READY  STATUS   RESTARTS  AGE
nginx1-744b889544-52dbg  1/1   Running  0      111s
nginx1-744b889544-6l4s9  1/1   Running  0      111s
nginx1-744b889544-cgzlr  1/1   Running  0      111s
nginx1-744b889544-w2gr7  1/1   Running  0      111s
nginx1-744b889544-zr5xz  0/1   Pending  0      111s
```

```
kubectl get pods -n namespace2
```

Expected output:

```
NAME          READY  STATUS   RESTARTS  AGE
nginx2-556f95449f-4gl8s  1/1   Running  0     111s
nginx2-556f95449f-crwk4  1/1   Running  0     111s
nginx2-556f95449f-gg6q2  0/1   Pending  0     111s
nginx2-556f95449f-pnz5k  1/1   Running  0     111s
nginx2-556f95449f-vjpmq  1/1   Running  0     111s
```

- This Deployment is similar to the `nginx1` Deployment. The total CPU resource requests of the pods in `namespace2` exceed `10 CPU cores`, which is the minimum number of CPU cores for `root.a.2`. The maximum number of CPU cores for the root is `40`. Therefore, these pods can use idle CPU cores in the cluster. The maximum number of CPU cores used by these pods cannot exceed `20`, which is the maximum number of CPU cores for `root.a.2`.
- If the number of CPU cores used by the pods in namespace2 reaches `20`, the next pod to be scheduled becomes pending. Among the five pods of this Deployment, four are in the running state and one is in the pending state.
- After you create the preceding two Deployments, the pods in `namespace1` and `namespace2` have used `40` CPU cores, which is the maximum number of CPU cores for the `root`.

8. Create a third Deployment in `namespace3` by using the following YAML template. This Deployment runs five pods and each pod requests five CPU cores.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx3
 namespace: namespace3
 labels:
  app: nginx3
spec:
 replicas: 5
 selector:
  matchLabels:
   app: nginx3
 template:
  metadata:
   name: nginx3
   labels:
    app: nginx3
  spec:
   containers:
   - name: nginx3
    image: nginx
    resources:
     limits:
      cpu: 5
     requests:
      cpu: 5
```

9. Run the following command to query the states of these pods:

```
kubectl get pods -n namespace1
```

Expected output:

```
NAME          READY  STATUS   RESTARTS  AGE
nginx1-744b889544-52dbg  1/1   Running  0     6m17s
nginx1-744b889544-cgzlr  1/1   Running  0     6m17s
nginx1-744b889544-nknns  0/1   Pending  0     3m45s
nginx1-744b889544-w2gr7  1/1   Running  0     6m17s
nginx1-744b889544-zr5xz  0/1   Pending  0     6m17s
```

```
kubectl get pods -n namespace2
```

Expected output:

```
NAME          READY  STATUS   RESTARTS  AGE
nginx2-556f95449f-crwk4  1/1   Running  0     4m22s
nginx2-556f95449f-ft42z  1/1   Running  0     4m22s
nginx2-556f95449f-gg6q2  0/1   Pending  0     4m22s
nginx2-556f95449f-hfr2g  1/1   Running  0     3m29s
nginx2-556f95449f-pvgrl  0/1   Pending  0     3m29s
```

```
kubectl get pods -n namespace3
```

Expected output:

```
NAME          READY  STATUS   RESTARTS  AGE
nginx3-578877666-msd7f  1/1   Running  0     4m
nginx3-578877666-nfdwv  0/1   Pending  0     4m10s
nginx3-578877666-psszr  0/1   Pending  0     4m11s
nginx3-578877666-xfsss  1/1   Running  0     4m22s
nginx3-578877666-xpl2p  0/1   Pending  0     4m10s
```

The `minimum number of CPU cores` for `root.b.1` is `10`. Therefore, when `nginx3` is created, the pods require the `minimum resources`. The scheduler reclaims the CPU cores that belong to `root.b` and are temporarily used by `root.a`. This ensures `a minimum of 10 CPU cores` for the pod scheduling of `nginx3`.

Before the scheduler reclaims the temporarily used 10 CPU cores, it also considers other factors, such as the priority classes, availability, and creation time of the workloads of `root.a`. Therefore, after the pods of `nginx3` are scheduled based on the reclaimed `10` CPU cores, two pods are in the running state and the other three are in the pending state.

10. Create a fourth Deployment `nginx4` in `namespace4` by using the following YAML template. This Deployment runs five pods and each pod requests five CPU cores.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx4
 namespace: namespace4
 labels:
  app: nginx4
spec:
 replicas: 5
 selector:
  matchLabels:
   app: nginx4
 template:
  metadata:
   name: nginx4
   labels:
    app: nginx4
  spec:
   containers:
   - name: nginx4
    image: nginx
    resources:
     limits:
      cpu: 5
     requests:
      cpu: 5
```

11. Run the following command to query the states of these pods:

```
kubectl get pods -n namespace1
```

Expected output:

```
NAME           READY  STATUS   RESTARTS  AGE
nginx1-744b889544-cgzlr  1/1   Running  0     8m20s
nginx1-744b889544-cwx8l  0/1   Pending  0     55s
nginx1-744b889544-gjkx2  0/1   Pending  0     55s
nginx1-744b889544-nknns  0/1   Pending  0     5m48s
nginx1-744b889544-zr5xz  1/1   Running  0     8m20s
```

```
kubectl get pods -n namespace2
```

Expected output:

```
NAME           READY  STATUS   RESTARTS  AGE
nginx2-556f95449f-cglpv  0/1   Pending  0     3m45s
nginx2-556f95449f-crwk4  1/1   Running  0     9m31s
nginx2-556f95449f-gg6q2  1/1   Running  0     9m31s
nginx2-556f95449f-pvgrl  0/1   Pending  0     8m38s
nginx2-556f95449f-zv8wn  0/1   Pending  0     3m45s
```

```
kubectl get pods -n namespace3
```

Expected output:

```
NAME          READY  STATUS   RESTARTS  AGE
nginx3-578877666-msd7f  1/1   Running  0     8m46s
nginx3-578877666-nfdwv  0/1   Pending  0     8m56s
nginx3-578877666-psszr  0/1   Pending  0     8m57s
nginx3-578877666-xfsss  1/1   Running  0     9m8s
nginx3-578877666-xpl2p  0/1   Pending  0     8m56s
```

```
kubectl get pods -n namespace4
```

Expected output:

```
nginx4-754b767f45-g9954  1/1   Running  0     4m32s
nginx4-754b767f45-j4v7v  0/1   Pending  0     4m32s
nginx4-754b767f45-jk2t7  0/1   Pending  0     4m32s
nginx4-754b767f45-nhzpf  0/1   Pending  0     4m32s
nginx4-754b767f45-tv5jj  1/1   Running  0     4m32s
```

Similarly, the `minimum number of CPU cores` for `root.b.2` is `10`. Therefore, when `nginx4` is created, the pods require the `minimum resources`. The scheduler reclaims the CPU cores that belong to `root.b` and are temporarily used by `root.a`. This ensures `a minimum of 10 CPU cores` for the pod scheduling of `nginx4`.

Before the scheduler reclaims the temporarily used 10 CPU cores, it also considers other factors, such as the priority classes, availability, and creation time of the workloads in `root.a`. Therefore, after the pods of `nginx4` are scheduled based on the reclaimed `10` CPU cores, two pods are in the running state and the other three are in the pending state.

After all the four Deployments are created, all pods in each namespace are running with the `minimum amount of resources` that are guaranteed for resource quotas.

17.5. Node scheduling

17.5.1. Use ECI elastic scheduling

Elastic Container Instance (ECI) elastic scheduling is an elastic scheduling strategy provided by Alibaba Cloud. You can add annotations to specify the resources that you want to use when you deploy applications. You can use only Elastic Compute Service (ECS) instances or elastic container instances, or automatically request elastic container instances when ECS resources are insufficient. ECI elastic scheduling can meet your resource requirements in different workload scenarios.

Prerequisites

- A professional Kubernetes cluster is created and the Kubernetes version is 1.18 or later. For more information, see Create a professional managed Kubernetes cluster.
- The **ack-virtual-node** component is installed in the cluster. For more information, see Use Elastic Container Instance in ACK clusters.

Procedure

Note You can add annotations to specify the type of resource that you want to use. The `alibabacloud.com/burst-resource` annotation can be set to one of the following values:

- If the value is left empty, only existing ECS resources in the cluster are used. This is the default value.
- **eci**: Elastic container instances are used when the ECS resources in the cluster are insufficient.
- **eci_only**: Only elastic container instances are used. The ECS resources in the cluster are not used.

1. Create the *nginx-deployment.yaml* file and copy the following content to the file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx
 labels:
  app: nginx
spec:
 replicas: 4
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   name: nginx
   annotations:
    alibabacloud.com/burst-resource: eci # This annotation specifies the type of resource that you want to u
se.
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx
    resources:
     limits:
      cpu: 2
     requests:
      cpu: 2
```

2. Run the following command to create pods that use elastic container instances:

```
kubectl apply -f nginx-deployment.yaml
```

18.Namespace and Quotas

18.1. Manage namespaces

You can create namespaces as workspaces to complete different tasks. This topic describes how to create, modify, and delete namespaces in a Kubernetes cluster.

Prerequisites

创建Kubernetes托管版集群

Context

In a Kubernetes cluster, you can use namespaces to create multiple virtual clusters. If multiple users share a cluster, you can create namespaces as workspaces to complete different tasks and use resource quotas to allocate resources in the namespaces.

Create a namespace

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, click **Namespaces and Quotas**.
5. On the page that appears, click **Create** in the upper-right corner.
6. In the **Create Namespace** dialog box, set the parameters. Then, click **OK**.
   - **Name**: Enter a name for the namespace. In this example, the name is set to test. The name must be 1 to 63 characters in length and can contain only digits, letters, and hyphens (-). The name must start and end with a letter or digit.
   - **Label**: Add one or more labels to the namespace. Labels are used to identify namespaces. For example, you can label a namespace as one used in the test environment.
     To add a label, enter a key and a value and click **Add** in the Actions column.

   Return to the Namespace page. In the list of namespaces, you can view that the namespace named test is created.

Edit a namespace

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, click **Namespaces and Quotas**.
5. On the **Namespace** page, find the namespace that you want to modify and click **Edit** in the **Actions** column.
6. In the dialog box that appears, click **Edit** to modify the label of the namespace. For example, modify the key-value pair of the label to `env:test-V2`. Then, click **OK**.

Delete a namespace

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.

3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, click **Namespaces and Quotas**.
5. Find the namespace that you want to delete and click **Delete** in the **Actions** column.
6. In the **Note** message, click **Confirm**.
   Return to the Namespace page. You can find that the namespace is deleted. Resource objects in the namespace are also deleted.

18.2. Set resource quotas and limits

You can set resource quotas and limits for a namespace in the Container Service for Kubernetes (ACK) console.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- A namespace is created. For more information, see Manage namespaces.
- You are connected to a master node of the cluster. For more information, see Connect to Kubernetes clusters by using kubectl.

Context

By default, pods that are in the running state can consume the CPU and memory resources of nodes without limit. In this case, pods in a namespace may exhaust the resources of the cluster.

Namespaces can be used as virtual clusters to serve multiple purposes. We recommend that you set resource quotas for namespaces.

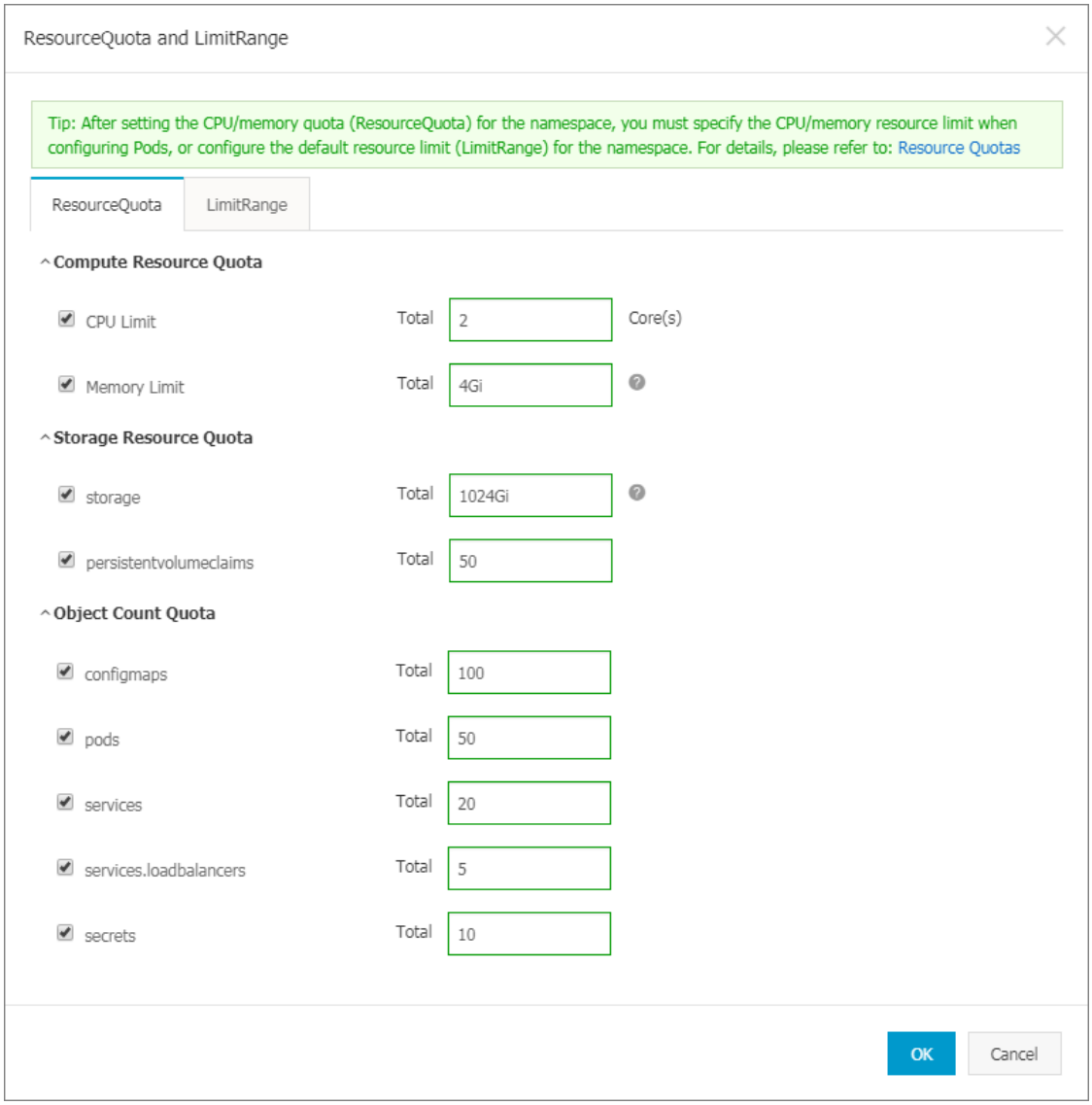
You can set multiple resource quotas for a namespace, including CPU, memory, and pod resource quotas. For more information, see Resource quotas.

Procedure

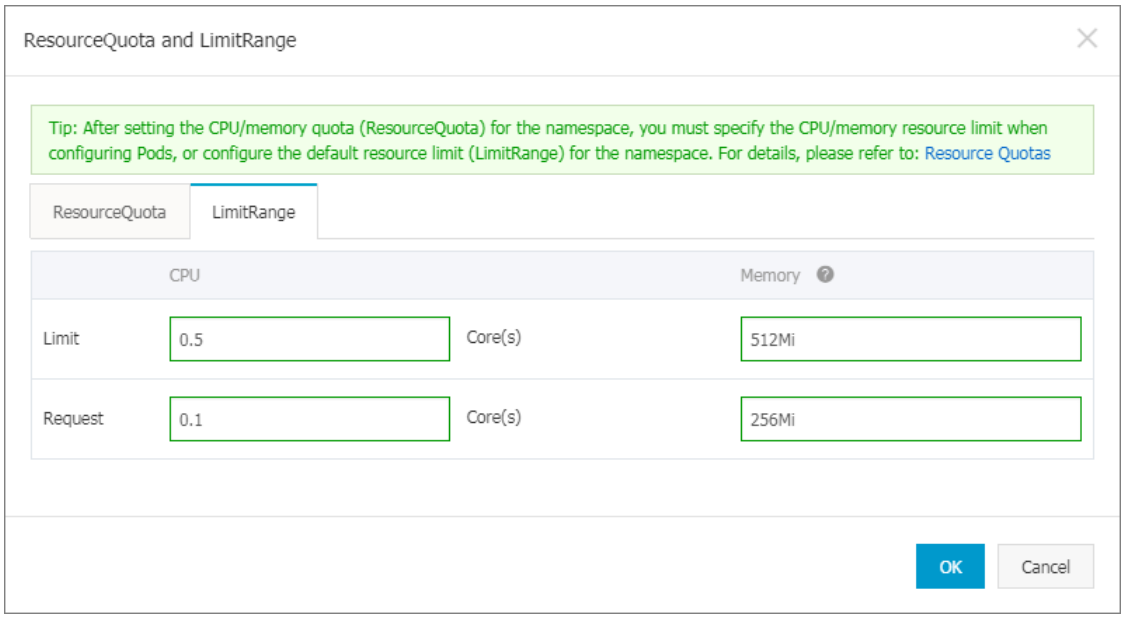
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, click **Namespaces and Quotas**.
5. Find the namespace that you want to manage and click **Resource Quotas and Limits** in the **Actions** column.
6. In the Resource Quotas and Limits dialog box, set resource quotas and default resource limits.

> **Note** Assume that you have set CPU or memory resource quotas for a namespace. When you create a pod, you must specify the CPU or memory limit for the pod or configure default limit ranges for the namespace. For more information, see Resource quotas.

i. You can set resource quotas for the namespace.



ii. You can set resource limits and resource requests for containers in the namespace. This enables you to control the amount of resources consumed by the containers. For more information, see Configure default memory requests and limits for a namespace.



7. Connect to a master node of the cluster and view the resources in the namespace.
   - Run the following command to query the resource quotas and limits:

     ```
     kubectl get limitrange,ResourceQuota -n test
     ```

     Expected output:

     ```
     NAME AGE
     limitrange/limits 8m
     NAME AGE
     resourcequota/quota 8m
     ```

   - Run the following command to query the resource quotas and limits:

     ```
     kubectl describe limitrange/limits resourcequota/quota -n test
     ```

     Expected output:

```
Name: limits
Namespace: test
Type Resource Min Max Default Request Default Limit Max Limit/Request Ratio
---- -------- --- --- --------------- ------------- -----------------------
Container cpu - - 100m 500m -
Container memory - - 256Mi 512Mi -
Name: quota
Namespace: test
Resource Used Hard
-------- ---- ----
configmaps 0 100
limits.cpu 0 2
limits.memory 0 4Gi
persistentvolumeclaims 0 50
pods 0 50
requests.storage 0 1Ti
secrets 1 10
services 0 20
services.loadbalancers 0 5
```

19.Disaster recovery center

19.1. Install the application backup component

Container Service for Kubernetes (ACK) allows you to back up and restore stateful applications deployed in an ACK cluster. This is an all-in-one solution to achieve crash consistency, application consistency, and cross-region disaster recovery for stateful applications in ACK clusters. You can install the application backup component in ACK clusters and then use the component to back up and restore applications. This topic describes how to install the application backup component.

Prerequisites

- The application backup feature is in public preview. To use this feature, you must Submit a ticket to apply to be added to the whitelist.
- A Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群, Create a dedicated Kubernetes cluster, or Register an external Kubernetes cluster.

  > **Notice** The Kubernetes version of your cluster must be later than 1.10. Otherwise, the feature cannot function as expected.

- Connect to an ACK cluster by using kubectl

Context

A growing number of applications are running on Kubernetes. Therefore, it is important to back up applications periodically. You can use backups to restore applications that cannot be recovered after the applications are disrupted for a long period of time. Traditional backup solutions include single-server backups and disk backups. Compared with the traditional backup solutions, application backups allow you to back up applications and the relevant data, resource objects, configurations, and namespaces.

Step 1: Grant OSS permissions to the cluster

The application backup feature can store application backups only in Object Storage Service (OSS). Before you use OSS, you must grant OSS permissions to your cluster.

Grant OSS permissions to managed or dedicated Kubernetes clusters

If you use a managed or dedicated Kubernetes cluster, you must grant the required OSS permissions to the cluster.

1. Create a custom permission policy that is used to access OSS. For more information, see Create a custom policy.

   > **Note** For more information about how to configure fine-grained access to OSS, see Use RAM to manage OSS permissions.

   To grant full OSS permissions, create a permission policy based on the following template:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:PutObject",
        "oss:GetObject",
        "oss:DeleteObject",
        "oss:GetBucket",
        "oss:ListObjects",
        "oss:ListBuckets"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

To grant only read and write permissions on a specified OSS bucket, create a permission policy based on the following template:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:PutObject",
        "oss:GetObject",
        "oss:DeleteObject",
        "oss:GetBucket",
        "oss:ListObjects",
        "oss:ListBuckets"
      ],
      "Resource": [
        "acs:oss:*:*:mybackups",
        "acs:oss:*:*:mybackups/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Replace `mybackups` with the name of the OSS bucket that you want to use.

2. Grant permissions to the Resource Access Management (RAM) role of the managed Kubernetes cluster.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
   iv. On the details page of the cluster, click the **Cluster Resources** tab and view the **worker RAM role** of the cluster.
   v. Log on to the RAM console.

vi. In the left-side navigation pane, click **Grants**.

vii. On the **Grants** page, click **Grant Permission**. In the **Add Permissions** panel, set the parameters and click **OK**.

Parameter	Description
Authorized Scope	Valid values: **Alibaba Cloud Account** and **Specific Resource Group**.
Principal	Enter the worker RAM role that you obtained.
Select Policy	Click **Custom Policy**, enter the name of the permission policy that is created in Step 1, and then click the name of the policy.

Grant OSS permissions to a registered Kubernetes cluster

If your applications are deployed in a registered Kubernetes cluster, you must create a RAM user for the cluster, grant the RAM user the permissions to access cloud resources, and then create an AccessKey pair for the RAM user.

1. Create a RAM user. For more information, see Create a RAM user.
2. Create a custom permission policy that is used to access OSS. For more information, see Step 1.
3. Grant permissions to the RAM user. For more information, see Grant permissions to a RAM user.
4. Create an AccessKey pair for the RAM user. For more information, see Obtain an AccessKey pair.
5. Create a Secret in the registered Kubernetes cluster.

   To ensure that the AccessKey pair is used only within the registered cluster, you must use the AccessKey pair to create a Secret named **alibaba-addon-secret** in the cluster. This reduces the risk of information leakage.
   ACK installs migrate-controller in the **velero** namespace, which is inherited from the open source Velero project. If the **velero** namespace does not exist, you must create a namespace named velero. After you create the namespace, use the AccessKey pair to create a Secret named alibaba-addon-secret in the namespace.

   i. (Optional)Run the following command to create a namespace named **velero**:

   ```
   kubectl create ns velero
   ```

   ii. Run the following command to create a Secret named alibaba-addon-secret:

   ```
   kubectl -n velero create secret generic alibaba-addon-secret --from-literal='access-key-id=<your AccessKey ID>' --from-literal='access-key-secret=<your AccessKey Secret>'
   ```

   Replace *your AccessKey ID* and *your AccessKey Secret* with the AccessKey ID and AccessKey secret that are obtained in Step 4.

Step 2: Install the application backup component

Note Before you can use the application backup feature, you must install the application backup component. If the application backup component is already installed, skip this step.

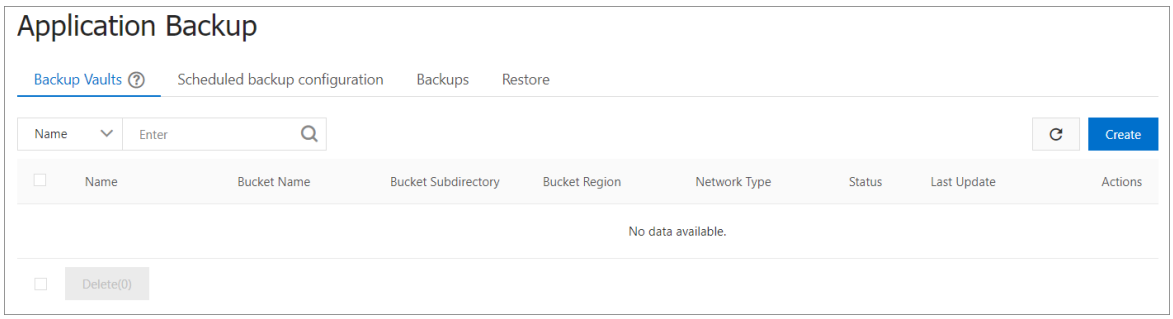
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or

click **Details** in the **Actions** column. The details page of the cluster appears.

4. In the left-side navigation pane of the details page, choose **Operations > Application Backup (Public Preview)**.
5. On the **Application Backup** page, click **Install**.

> **Note** If the velero namespace does not exist, the system automatically creates a namespace named velero when the system installs the component. Do not delete this namespace when you back up applications.

After the component is installed, the page in the following figure appears.



Related information

- Use migrate-controller to back up and restore applications
- Migrate applications across clusters

19.2. Use migrate-controller to back up and restore applications

The migrate-controller component is developed by Alibaba Cloud based on the open source project Velero. You can use this component to migrate Kubernetes applications. This topic describes how to use migrate-controller to back up and restore applications in a Container Service for Kubernetes (ACK) cluster.

Prerequisites

Install the application backup component

Back up applications

You can use the Velero tool of migrate-controller to back up all resources in one or more specified namespaces of an ACK cluster. Command:

```
velero backup create <BACKUP NAME> --include-namespaces <INCLUDE NAMESPACES>
```

The following examples show how to back up all resources in one namespace and all resources in two namespaces by running commands:

- Back up applications in the nginx-examples namespace:

  ```
  velero backup create backup-nginx --include-namespaces nginx-examples
  ```

- Back up applications in the nginx-examples namespace and wordpress namespace:

  ```
  velero backup create backup-nginx-and-wordpress --include-namespaces nginx-examples,wordpress
  ```

> **Note** The preceding operations do not back up the data of persistent volumes (PVs) that are mounted to the applications. Instead, only orchestration files in the JSON format are backed up.

If you want to back up the PVs, perform the following steps:

1. Add the backup.velero.io/backup-volumes annotation to pods to which the PVs are mounted. Command:

   ```
   kubectl -n <NAMESPACE> annotate pod/<POD NAME> backup.velero.io/backup-volumes=<POD VOLUME NAME>
   ```

2. After you add the annotation, back up the PVs. Command:

   ```
   velero backup create <BACKUP NAME> --include-namespaces <INCLUDE NAMESPACES>
   ```

   For example, you want to back up an NGINX application in the nginx-examples namespace and the PV of the application. The PV is named nginx-logs and mounted to the nginx-deployment-7477779c4f-rz95l pod. To back up the PV, perform the following steps:

   i. Add an annotation to the pod to which the PV is mounted. Command:

   ```
   kubectl -n nginx-example annotate pod/nginx-deployment-7477779c4f-rz95l backup.velero.io/backup-volumes=nginx-logs
   ```

   ii. Back up the PV. Command:

   ```
   velero backup create nginx-backup-with-pv --include-namespaces nginx-example
   ```

Back up applications at a scheduled time

To back up applications at a scheduled time, run the following command to create a scheduled backup task:

```
velero schedule create <NAME> --include-namespaces <NAMESPACE> --schedule "SCHEDULE"
```

The following examples show how to back up applications every 7 days and every 24 hours by running commands:

- Create the nginx-app-with-pv-schedule-01 scheduled task to back up applications in the nginx-examples namespace every 7 days:

  ```
  velero schedule create nginx-app-with-pv-schedule-01 --include-namespaces nginx-examples --schedule "0 7 * * *"
  ```

- Create the nginx-app-with-pv-schedule-02 scheduled task to back up applications in the nginx-examples namespace every 24 hours:

  ```
  velero schedule create nginx-app-with-pv-schedule-01 --include-namespaces nginx-examples --schedule "@every 24h"
  ```

Restore applications

This example describes how to restore applications after the namespace to which the applications belong is deleted.

1. Run the following command to delete the nginx-example namespace:

   ```
   kubectl delete ns nginx-example
   ```

2. Run the following command to restore applications:

```
velero restore create --from-backup nginx-backup-without-pv
```

Related information

- Migrate applications across clusters

19.3. Migrate applications across clusters

Container Service for Kubernetes (ACK) allows you to back up and restore stateful applications deployed in an ACK cluster. This is an all-in-one solution to achieve crash consistency, application consistency, and cross-region disaster recovery for stateful applications in ACK clusters. This topic describes how to migrate applications across clusters by using the application backup feature or Velero.

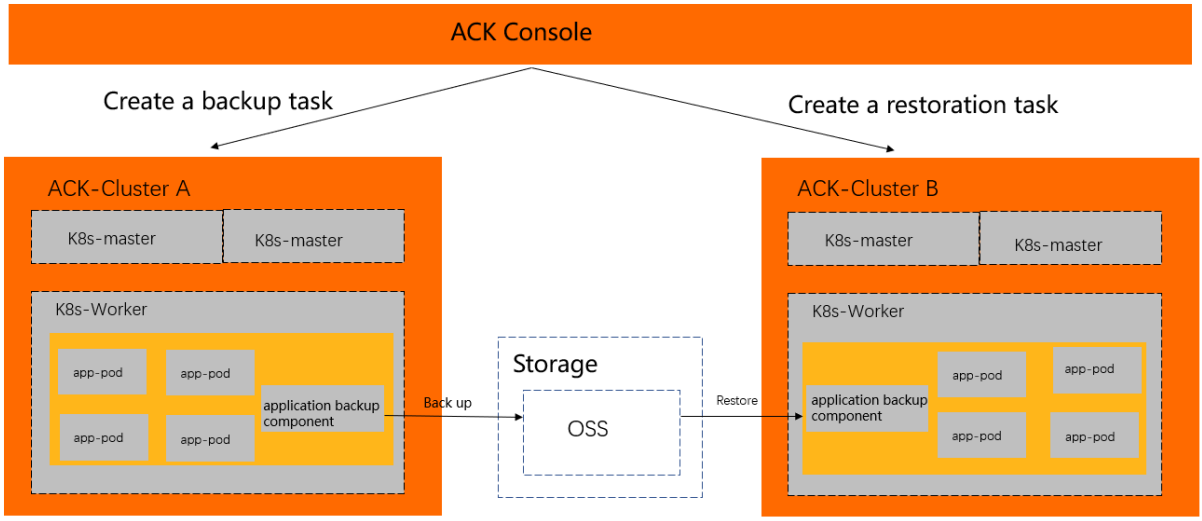
Prerequisites

Two Kubernetes clusters of the same version are created. For more information, see 创建Kubernetes托管版集群.

Introduction

In this example, a cluster named Cluster_A and a cluster named Cluster_B are used. The following example shows how to back up an application in Cluster_A and restore the application to Cluster_B.

Notice Make sure that the two clusters run the same Kubernetes version. Otherwise, you cannot migrate the application to the specified cluster.



Method 1: Migrate an application by using the application backup feature

1. Grant the required Object Storage Service (OSS) permissions to Cluster_A and Cluster_B. For more information, see Configure OSS permissions.
2. Install the application backup component in Cluster_A and Cluster_B. For more information, see Install the application backup component.
3. Create backup vaults for Cluster_A and Cluster_B. When you create backup vaults for Cluster_A and Cluster_B, select the same OSS bucket and set Network Type to Public Network. For more information, see Create a backup vault.
4. Create a backup task in Cluster_A. For more information, see Create a backup task.

   After you create the backup task in Cluster_A, you can log on to Cluster_B to view the backup task on

the **Backups** tab of the **Application Backup** page.

5. On the **Application Backup** page, click the **Restore** tab and create a restoration task based on the backup task that is created in Cluster_A. For more information, see Create a restoration task.

Method 2: Migrate an application by using Velero

This example shows how to migrate an application in the nginx-examples namespace from Cluster_A to Cluster_B. To migrate the application, perform the following steps:

1. Install the Velero client and the application backup component in Cluster_A. Then, configure and connect to the OSS bucket named mybackups. For more information, see velero and Install the application backup component.
2. Install the Velero client and migrate-controller in Cluster_B. Then, configure and connect to the OSS bucket named **mybackups**.
3. Back up applications in the **nginx-examples** namespace in Cluster_A and specify the name of the backup file, for example, **nginx-backup**.
4. Use the backup file named **nginx-backup** in Cluster_B to restore the applications. This operation migrates the applications from Cluster_A to Cluster_B.

In this example, an application in Cluster_A is backed up and then restored to Cluster_B. This way, the application is migrated from Cluster_A to Cluster_B. Before you back up and restore the application, you must complete the operations in Step 1 and Step 2. The following example shows how to back up and restore the application:

1. Deploy the sample application named nginx-app-with-pv in Cluster_A.
   i. Create a file named *nginx-app-with-pv.yaml* and copy the following content to the file:

```
---
apiVersion: v1
kind: Namespace
metadata:
 name: nginx-example
 labels:
  app: nginx
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: nginx-logs
 namespace: nginx-example
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
   storage: 20Gi
 storageClassName: alicloud-disk-ssd
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
 namespace: nginx-example
spec:
 replicas: 1
 template:
  metadata:
```

```
  metadata:
    labels:
      app: nginx
    annotations:
      pre.hook.backup.velero.io/container: fsfreeze
      pre.hook.backup.velero.io/command: '["/sbin/fsfreeze", "--freeze", "/var/log/nginx"]'
      post.hook.backup.velero.io/container: fsfreeze
      post.hook.backup.velero.io/command: '["/sbin/fsfreeze", "--unfreeze", "/var/log/nginx"]'
  spec:
    volumes:
      - name: nginx-logs
        persistentVolumeClaim:
          claimName: nginx-logs
    containers:
    - image: nginx:1.7.9
      name: nginx
      ports:
      - containerPort: 80
      volumeMounts:
        - mountPath: "/var/log/nginx"
          name: nginx-logs
          readOnly: false
    # sync from gcr.io/heptio-images/fsfreeze-pause:latest
    - image: registry.cn-hangzhou.aliyuncs.com/acs/fsfreeze-pause:latest
      name: fsfreeze
      securityContext:
        privileged: true
      volumeMounts:
        - mountPath: "/var/log/nginx"
          name: nginx-logs
          readOnly: false
```

ii. Run the following command to deploy the application to Cluster_A:

```
kubectl apply -f nginx-app-with-pv.yam
```

2. Run the following commands to view information about the application.
   - Run the following command to query the Deployment:

     ```
     kubectl -n nginx-example get deployment
     ```

     Expected output:

     ```
     NAME                  READY  UP-TO-DATE  AVAILABLE  AGE
     deployment.extensions/nginx-deployment  1/1   1      1     108s
     ```

   - Run the following command to view the pod:

     ```
     kubectl -n nginx-example get pod
     ```

     Expected output:

     ```
     NAME                  READY  STATUS   RESTARTS  AGE
     pod/nginx-deployment-7477779c4f-rz95l  2/2   Running  0     108s
     ```

   - Run the following command to query information about the PVC:

     ```
     kubectl -n nginx-example get pvc
     ```

Expected output:

```
NAME                STATUS  VOLUME         CAPACITY  ACCESS MODES  STORAGECLASS     AGE
persistentvolumeclaim/nginx-logs  Bound  d-bp1eutshfe774exa5os2  20Gi  RWO  alicloud-disk-ssd  108s
```

3. Back up the namespace to which the application belongs.
   - Run the following command to back up the namespace to which the application belongs (excluding PV data):

     ```
     velero backup create nginx-backup-without-pv --include-namespaces nginx-example
     ```

   - Run the following command to back up the namespace to which the application belongs (including PV data).
     a. Run the following command to add annotations to the pod volume that needs to be backed up:

        ```
        kubectl -n nginx-example annotate pod/nginx-deployment-7477779c4f-rz95l backup.velero.io/backup-volumes=nginx-logs
        ```

     b. Run the following command to back up the application:

        ```
        velero backup create nginx-backup-with-pv --include-namespaces nginx-example
        ```

4. (Optional)Run the following command to view information about the backup file:

   ```
   velero backup get
   ```

   Expected output (excluding PV data):

   ```
   NAME                  STATUS     ERRORS  WARNINGS  CREATED                         EXPIRES  STORAGE LOCATION  SELECTOR
   nginx-backup-without-pv  Completed  0       0         2020-09-18 14:43:13 +0800 CST  29d      default           <none>
   ```

   The output indicates that a backup file is created for the application.

5. Run the following command to restore the application to Cluster_B (excluding PV data):

   ```
   velero restore create --from-backup nginx-backup-without-pv
   ```

20.Application center

20.1. Overview

Application center is a declarative, GitOps continuous delivery tool for Kubernetes based on Argo CD. In GitOps pattern, application definitions, configurations are declarative and version controlled, application deployment and lifecycle management are automated and easy to understand. This topic describes the features, architecture, concepts, and core components of **Application Center**.

Features

The application center follows the GitOps pattern of using Git repositories as the source of truth for definning the desired application state. It continuously monitors running applications and compares the current live state against the desired target state(specified in the Git repo). Application center reports & visualizes the differences, while providing facilities to automatically or manually sync the live state back to the desired target state. Any modifications made to the desired target state in the Git repo can be automatically applied and reflected in the specified target environments.

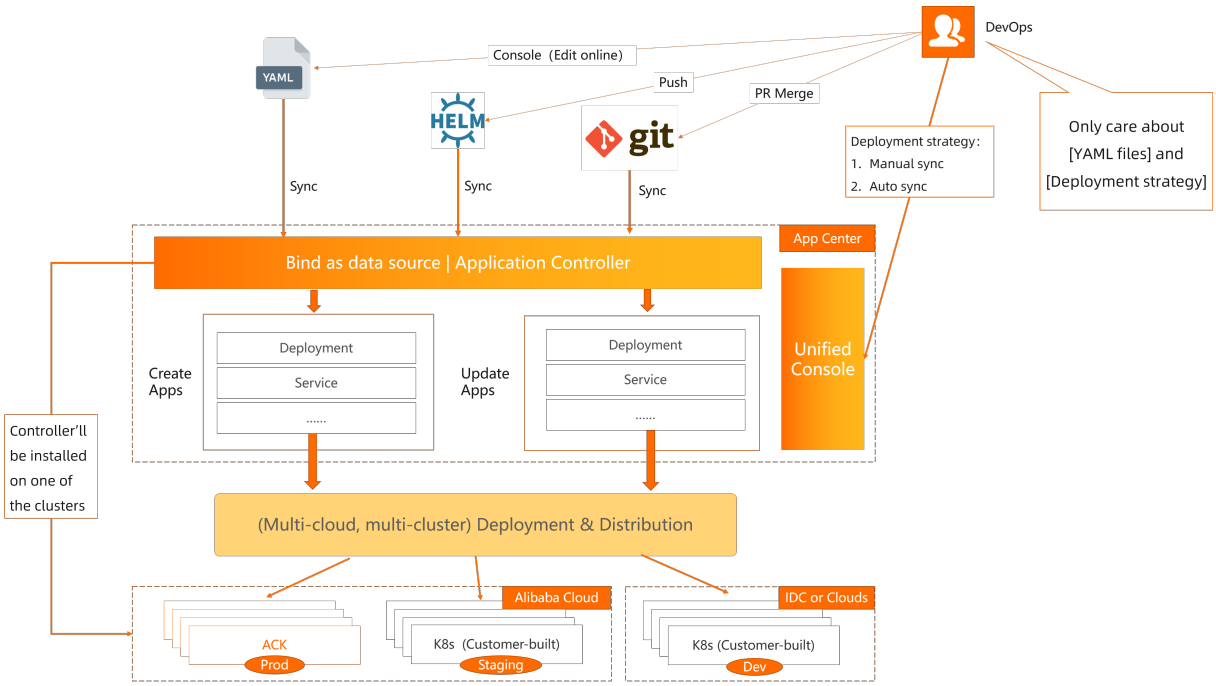
Application center features list as below:

- Automated deployment of applications to specified target clusters from Git repo or Helm OCI artifact repo.
- Ability to manage and deploy to multiple clusters(public cloud by ACK, edge computing, on-premises and other clouds).
- Health status analysis of application resources.
- Automated or manual syncing of applications to its desired state.
- Rollback/Roll-anywhere to any application configuration committed in Git repository.
- Web UI which provides real-time view of application activity.
- Webhook integration (Git, ACREE).

Architecture

The application center enables centralized management of applications. This allows you to view the overall topologies of applications in your cluster. Application Center is applicable to the following scenarios:

- You can view the deployment status and changes of all Kubernetes resources used by each application.
- You can also use templates or Helm charts on GitHub to deploy different versions of an application to a cluster.
- You can roll back the application to each version or deploy the application versions to a cluster.



Concepts

Concept	Description
Application	A set of ACK resources created based on an orchestration template.
Target state	The target state of an application is defined by files in repositories such as Gits or Helm charts.
Current state	The up-to-date state of an application, such as the states of all pods.
Deployment status	Whether the up-to-date state of an application is the same as the target state. Whether the state of a deployed application is the same as that defined by the Gits or Helm charts.
Deployment	The process of changing an application from the current state to a target state. For example, it may refer to the process where you perform **kubectl apply** to deploy an application in an ACK cluster.
Refresh	Compare the up-to-date state with the state as defined in the latest code in Gits.
Health condition	Whether the application is running properly.

Core components

All components of Application Center are deployed in the **appcenter** namespace of your cluster. The following table lists the four core components.

Component	Description
application-controller	Records and backtracks application versions and rolls back an application to a specified version.
redis	Caches application data and records the cached application data.

Component	Description
repo-server	Pulls deployment templates from remote repositories, such as Gits or Helm charts.
server	Makes a gRPC server accessible over the Internet. The gRPC server is used to receive and process external requests.

Related information

-
- Deploy an application in Application Center

20.2. Quick start

This topic uses a video to describe how to use the features of the **application center** in the Container Service for Kubernetes (ACK) console.

Related information

- Overview

20.3. Install the Application Center controller

Before you deploy an application in Application Center, you must install the Application Center controller. The Application Center controller enables Application Center. After the controller is installed, you can deploy applications and manage application versions in Application Center. This topic describes how to install the Application Center controller.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center (Previous Version)**.
3. Install the Application Center controller.

   Select the cluster in which you want to install the controller and click **Install**. It requires about three minutes to complete the installation process.
4. Add a cluster to Application Center.

   By default, the system adds the cluster where the Application Center controller is installed to Application Center. After you add a cluster, you can deploy applications to the added cluster in Application Center. The Application Center controller is installed in only the cluster that you select when you initialize Application Center. You do not need to install the controller in the added clusters. This provides an easy method for you to deploy and manage applications in multiple clusters.

   After you add a cluster, click **Get Started** to deploy applications.

Related information

- Overview
- Deploy an application in Application Center

20.4. Configuration management

20.4.1. Configure a certificate

Before you connect to a repository that supports HTTPS connections, you must configure a certificate for connecting to the repository. This topic describes how to add a TLS certificate and an SSH known host to the application center.

Add a TLS certificate

1. Log on to the ACK console.
2. Click **ADD TSL CERTIFICATE**. On the **Create TLS repository certificate** page, specify **Repository server name** and **TLS certificate (PEM format)**.

   > **Notice** The repository server name and the TLS certificate must be in the PEM format. Otherwise, the certificate creation fails.

3. Click **Create**.

Add an SSH known host

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Application Center > Configuration** and click **Certificates**. The **Repository certificates** page appears.
3. Click **ADD SSH KNOWN HOSTS**. On the **Create SSH known host entries** page, specify **SSH known hosts data**.
4. Click **CREATE**.

Related information

- Connect to a repository

20.4.2. Connect to a repository

After you connect to a repository, you can select it when you create an application. This topic describes how to use SSH or HTTPS to connect to a repository.

Use SSH to connect to a repository

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Application Center > Configuration** and click **Repositories**. The **Repositories** page appears.
3. Click **CONNECT REPO USING SSH**. On the **Connect repo using SSH** page, configure the parameters.

Parameter	Description
Name (required for Helm)	The name of the repository. You can specify a custom name.
Repository URL	The URL of the repository. **Note** You must specify a repository URL that can be accessed from the Internet.

Parameter	Description
SSH private key data	The SSH private key for identity verification. You can connect to the repository only after you pass the identity verification.
Skip server verification	Specifies whether to skip the GitHub server verification.
Enable LFS support (Git only)	Specifies whether to enable Git Large File Storage (LFS). If you enable this feature, you can store large files in the GitHub repository that you connect to.

4. Click **CONNECT**.

Use HTTPS to connect to a repository

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Application Center > Configuration** and click **Repositories**. The **Repositories** page appears.
3. Click **CONNECT REPO USING HTTPS**. On the **Connect repo using HTTPS** page, configure the parameters.

Parameter	Description
Type	The type of the repository. Select **git** or **helm**.
Repository URL	The URL of the repository. **Note** You must specify a repository URL that can be accessed from the Internet.
Username (optional)	The user name and password for identity verification. You can connect to the repository only after you pass the identity verification.
Password (optional)	
TLS client certificate (optional)	The TLS client certificate and certificate key for identity verification. You can connect to the repository only after you pass the identity verification.
TLS client certificate key (optional)	
Skip server verification	Specifies whether to skip the GitHub server verification.
Enable LFS support (Git only)	Specifies whether to enable Git LFS. If you enable this feature, you can store large files in the GitHub repository that you connect to.

4. Click **CONNECT**.

Related information

- Configure a certificate

20.4.3. Configure a cluster

After you configure a cluster, you can select it when you create an application. This topic describes how to configure a cluster.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Application Center > Configuration** and click **Clusters**. The **Clusters** page appears.
3. Click **ADD CLUSTER** and select the target cluster.
4. Click **Add**.

20.5. Application management

20.5.1. Deploy an application in Application Center

You can view the deployment state and changes of all Kubernetes resources when you deploy an application in Application Center in the Container Service for Kubernetes (ACK) console. You can choose different methods to deploy applications based on the data sources of the templates. Application Center allows you to use templates from Git repositories or ACK orchestration templates in the ACK console. This topic describes how to use different templates to deploy applications.

Prerequisites

Before you deploy an application, make sure that you have completed the following steps:

- 创建Kubernetes托管版集群
- Install the Application Center controller

You can use Git repositories as data sources to maintain and manage Git templates or Helm charts.

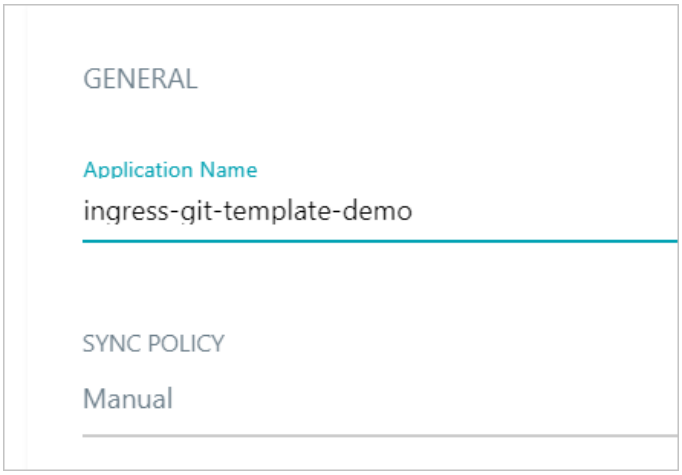
Use Git repositories as data sources to deploy an application from a YAML template

In this example, an application is deployed by using the ingress-demo YAML file from a Git repository. Perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center (Previous Version)**.
3. In the upper-right corner of the **Application** page, click **+ NEW APP** and set the parameters.

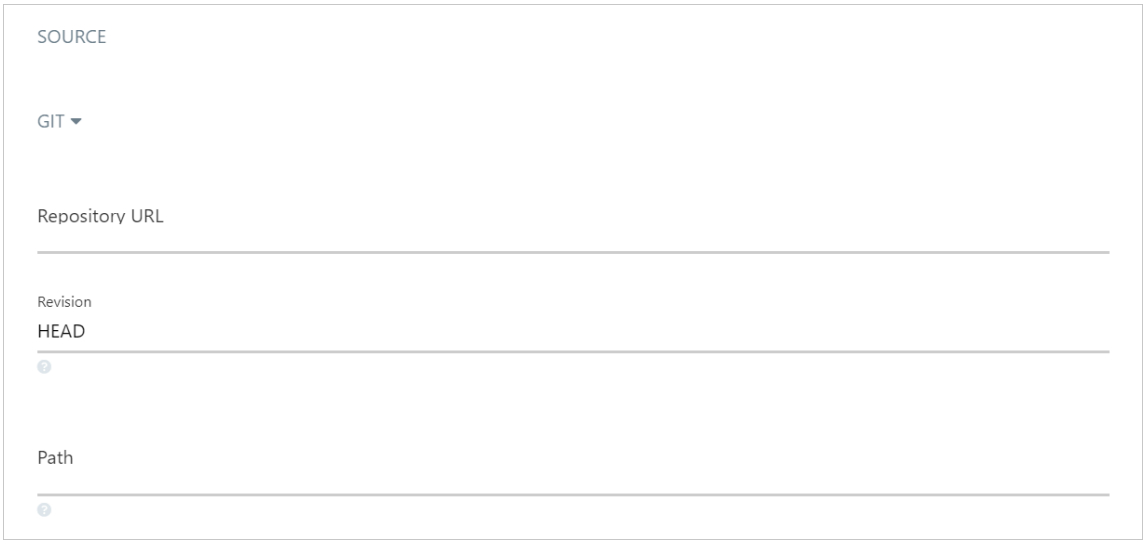
   The following tables describe the parameters.

   - General settings



Parameter	Description
Application Name	The name of the application. You can specify a custom name.
SYNC POLICY	The synchronization mode of the template. In this example, the manual mode is selected. Valid values: ■ Manual: If the template on the Git repository is updated, you must click **SYNC** in Application Center to manually update the template and deploy the latest version of the application. ■ Automatic: If the template in the Git repository is updated, ACK automatically updates the template and deploys the latest version of the application.

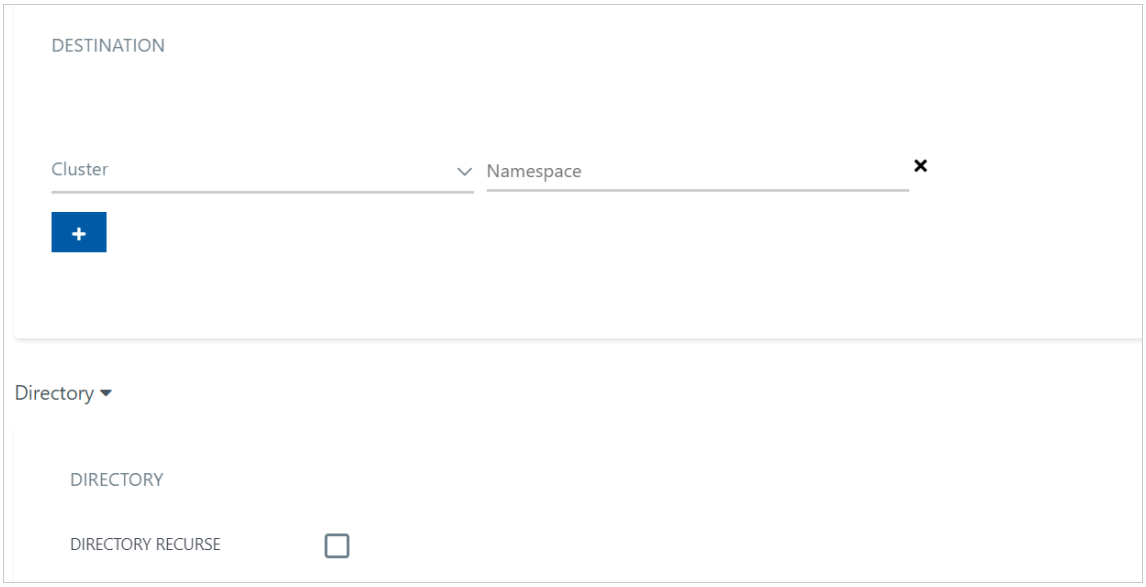
- Source settings



Parameter	Description
Data source	The data source that is used to deploy an application. Valid values: GIT, HELM, and Custom template. In this example, **GIT** is selected.

Parameter	Description
Repository URL	The URL that is used to download the template from a Git repository. We recommend that you set this parameter to a value that starts with HTTPS. In this example, the URL is set to https://github.com/xianlubird/argocd-example-apps.
Revision	The revision of the template. You can also specify a Git branch. In this example, the revision is set to HEAD. This indicates that the **master** branch is used. You can also specify other Git branches.
Path	The path where the template is stored. Set this parameter to a directory. In this example, the path is set to *ingress-demo*. **Notice** Do not set the path to the name of a file.

- Destination settings



Parameter	Description
Cluster	The cluster to which the application is deployed. In this example, the default cluster is selected. You can also select the cluster where you want to deploy the application.
Namespace	The namespace where the application is deployed.

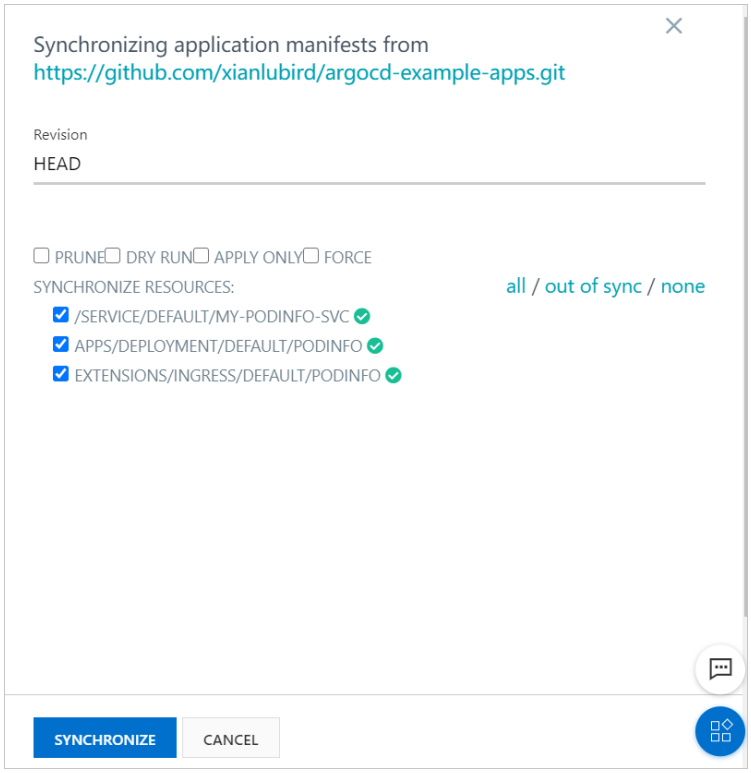
Parameter	Description
Directory/Helm/Kustomize	Select a format for the template. In this example, Directory is selected. ■ Directory: If you select Directory, you can specify whether to enable **directory recursion**. We recommend that you enable directory recursion if the path that stores the template has multiple levels. ■ Helm: ■ VALUES FILES: the file that contains the values to be specified. In this example, the default values.yaml file in the Helm chart is specified. ■ VALUES: specify values. If you specify values, the values referenced from the values.yaml is overwritten by the specified values. ■ Kustomize: ■ NAME PREFIX: the prefix of a template that you want to use under the management of Kustomize. You can specify this parameter to search for templates that are named with the specified prefix. ■ NAME SUFFIX: the suffix of a template that you want to use under the management of Kustomize. You can specify this parameter to search for templates that are named with the specified suffix.

4. Click **CREATE**.

   After the creation is complete, you can view the details of the application in Application Center.

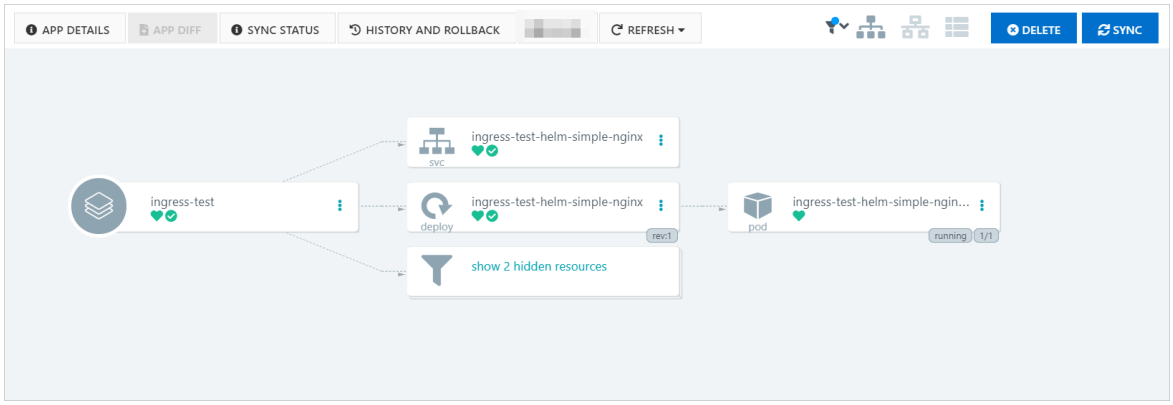
   > **Notice** After the creation is complete, Application Center parses the deployment template. However, the related resource objects are not deployed in the cluster. You can view the related resource objects in the deployment template.

5. In the upper-right corner of the details page of the application, click **SYNC**. In the dialog box that appears, select the resources that you want to deploy and click **SYNC**.



Parameter	Description
PRUNE	**Deletes** the application resources that are not specified in the Git template.
DRY RUN	Parses the template only. The application resources specified in the template are not deployed in the cluster.
APPLY ONLY	Deploys the related resource objects that are specified in the template in the cluster. This option is applicable to most scenarios.
FORCE	Forcibly aligns the selected resource objects to those specified in the template. If the you select more than those specified in the template, the additional resource objects are removed from the deployment. If you select fewer, the unselected resource objects are added to the deployment.
SYNCHRONIZE RESOURCES	Lists all resource objects specified in the template. You can deploy certain or all resource objects based on your requirements. By default, all resource objects are deployed. In this example, all resource objects are deployed.

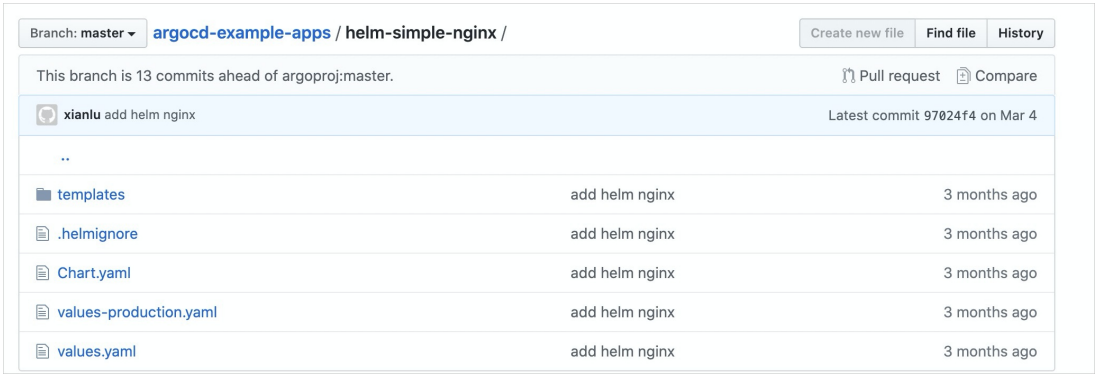
In the upper-left corner of the details page of the application, click **APP DETAILS** to view the details. The following figure shows the details page of an application.



Use Git repositories as data sources to deploy an application from a Helm chart

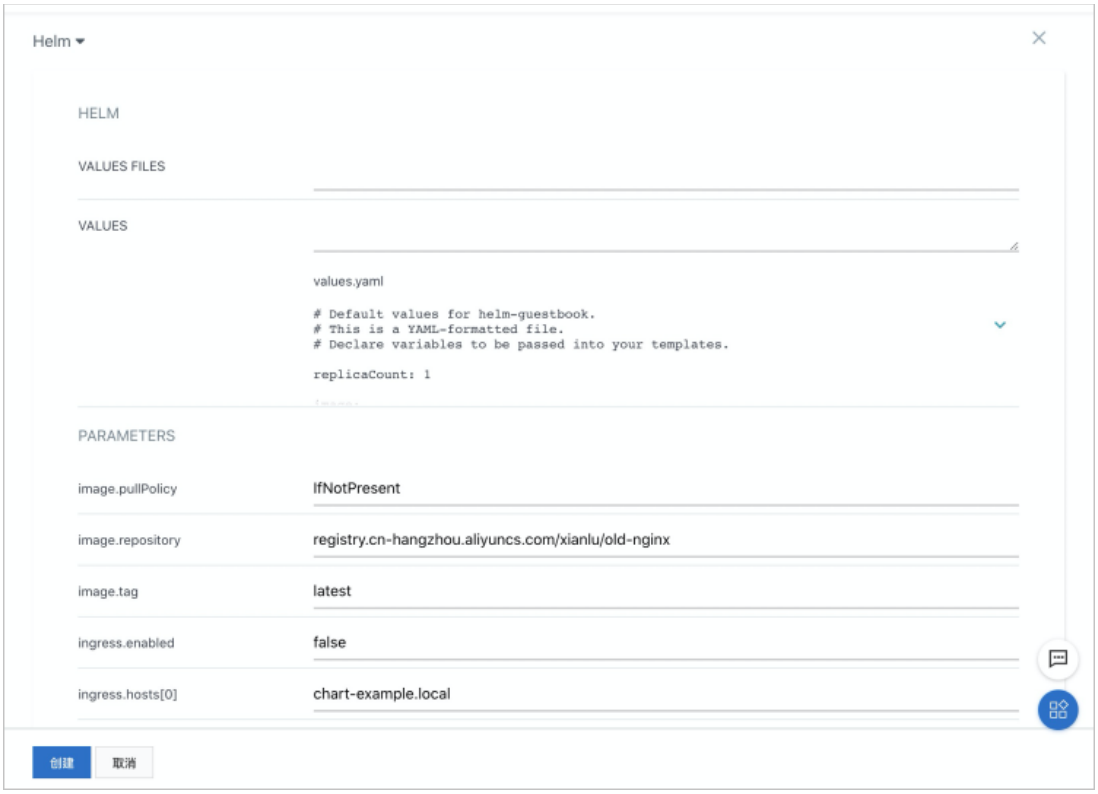
If you use a Helm chart from a Git repository to deploy an application, follow the steps as described in Use Git repositories as data sources to deploy an application from a YAML template.

A Helm chart



When you create an application, select **Helm** as the template format. You can set VALUES to different values for different clusters.

Helm chart parameters



Use an ACK orchestration template to deploy an application

You can also use an ACK orchestration template in the ACK console to deploy an application. Perform the following steps:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > Orchestration Templates**.
3. Select the template that you want to use and click **Create Application**.
4. In the **Deploy** section, click **Create** and then click **Save**.
5. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center (Previous Version)**.
6. Configure the parameters of the ACK orchestration template.

   When you create an application, select **Custom template** in the SOURCE section. Click the **Template** bar and select the template from the drop-down list. For more information about how to create an application, see Use Git repositories as data sources to deploy an application from a YAML template.

Related information

- Overview

20.5.2. View applications in Application Center

This topic describes how to view all applications or specify filter conditions to view specified applications in the Container Service for Kubernetes (ACK) console.

View all applications

Log on to the ACK console. In the left-side navigation pane, choose **Multi-cluster > Application Center**. On the **Application** page, you can view all deployed applications.

View details about an application

Log on to the ACK console. In the left-side navigation pane, choose **Multi-cluster > Application Center**. On the Application page, find and click the name of the application that you want to view. On the page that appears, click **APP DETAILS** to view application information, for example, you can click **SUMMARY**, **PARAMETERS**, **MANIFEST**, **DIFF**, and **EVENTS** to view the related information.

Filter applications

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center (Previous Version)**.
3. In the left-side section of the **Application** page, set filter conditions or enter an application name. The applications that match the filter conditions appear on the right side of the page.

20.5.3. Update an application in Application Center

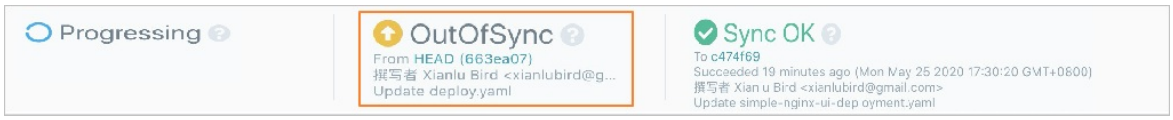
You can update an application in the Container Service for Kubernetes (ACK) console.

Prerequisites

- If you use a template from a Git repository to update an application, make sure that the template in the Git repository is updated to the latest version.
- If you use an ACK orchestration template to update an application, make sure the ACK orchestration template in the ACK console is updated to the latest version. For more information, see Modify an orchestration template.

Procedure

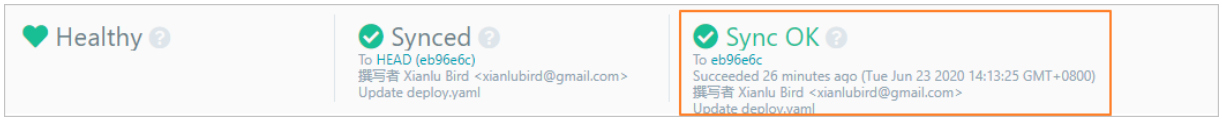
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center (Previous Version)**.
3. Click the name of the application to go to the details page of the application.
4. Click **REFRESH** and then click **Hard Refresh** to obtain the latest template.
   In the following figure, the application is in the **OutOfSync** state.



5. In the upper-right corner of the details page, click **SYNC**.
6. In the dialog box that appears, select the resource objects that you want to deploy and click **SYNCHRONIZE**.

Result

If the application changes to the **Sync OK** state, the application is updated.



20.5.4. Roll back an application in Application Center

You can roll back an application created in Application Center to an earlier version and view details about each application version. This topic describes how to roll back an application.

Prerequisites

Deploy an application in Application Center

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center (Previous Version)**.
3. Click the name of the application that you want to roll back. On the details page of the application, click **HISTORY AND ROLLBACK**.
4. On the page that appears, find the application version to which you want to roll back, click ⋮ in the upper-right corner of the page and click **Rollback**.
5. In the **Rollback application** message, click **OK**.
   After the rollback is complete, the application changes to the **Out Of Sync** state.

What's next

You can update templates in source repositories and deploy new application versions to a cluster. For more information, see Update an application in Application Center.

Related information

- View applications in Application Center

20.5.5. Delete an application from Application Center

This topic describes how to delete an application from Application Center.

Prerequisites

Deploy an application in Application Center

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center (Previous Version)**.
3. Click the name of the application that you want to delete. In the upper-right corner of the details page, click **DELETE**.
4. In the **Delete application** dialog box, click **OK**.

> **Notice** If you select **Cascade**, all sub-resources of the application are deleted.

20.6. Triggers

20.6.1. Overview

A trigger is a webhook that you can create for your applications in Application Center. You can deploy applications created in Application Center to an ACK cluster by using triggers in a third party environment.

Prerequisites

Triggers can be used to update or redeploy applications created in Application Center. You must create applications in Application Center before you can use triggers.

Scenarios

Triggers are applicable in the following scenarios:

- Use triggers in a third party environment to redeploy applications that have already been deployed in ACK clusters by Application Center.
- Use triggers to automatically connect Application Center to existing continuous integrate (CI) systems, such as a Container Registry delivery pipeline, Jenkins, and GitLab CI/CD. This forms a complete CI/CD pipeline.

Precautions

Keep your triggers safe. If a trigger is leaked, you must delete the trigger and create a new one.

Related topics

- Manage triggers
- Use triggers

20.6.2. Manage triggers

You can manage triggers in the Alibaba Cloud Container Service for Kubernetes (ACK) console. This topic describes how to create, view, and delete triggers.

Prerequisites

Deploy an application in Application Center

Create a trigger

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center (Previous Version)**.
3. On the **Application** page, click the target application section.
4. On the information page of the target application, click the **Triggers** tab.
5. On the **Triggers** page, click **CREATE**.

View a trigger

You can view the trigger that you have created for your applications in the ACK console. For more information, see Create a trigger.

Delete a trigger

On the **Triggers** page, click **Delete**. For more information, see Create a trigger.

20.6.3. Use triggers

You can use triggers in a third-party environment to deploy applications that are created in Application Center to clusters of Container Service for Kubernetes (ACK) or redeploy applications that have already been deployed in ACK clusters by Application Center. This topic describes how to use triggers.

Prerequisites

- Deploy an application in Application Center
- Connect to Kubernetes clusters by using kubectl

Context

You can create applications in Application Center from the following data sources: Git repositories, Helm repositories, and ACK orchestration templates.

- If an application is created from an ACK orchestration template, you can use triggers to update the image information (such as the tag of the image) in the template, pull the updated template, and use it to redeploy the application.
- If an application is created from a Git or Helm repository, you can only pull the latest template and use the template to redeploy the application. You cannot use triggers to update the template.

In this example, an application named demo is created. It runs on a Deployment named demo. The URL of the trigger that is created for the application is in the following format: `https://cs.console.aliyun.com/hook/trigger?token=xxxxxxxxx`.

Use the following template to create the demo Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: demo
 labels:
  app: demo
spec:
 minReadySeconds: 5
 revisionHistoryLimit: 5
 progressDeadlineSeconds: 60
 strategy:
  rollingUpdate:
   maxUnavailable: 1
  type: RollingUpdate
 selector:
  matchLabels:
   app: demo
 template:
  metadata:
   annotations:
    prometheus.io/scrape: "true"
    prometheus.io/port: "9797"
   labels:
    app: demo
  spec:
   containers:
   - name: demo
    image: registry.cn-hangzhou.aliyuncs.com/acs/rollouts-demo:blue
    imagePullPolicy: IfNotPresent
    ports:
    - name: http
     containerPort: 8080
     protocol: TCP
    readinessProbe:
     tcpSocket:
      port: 8080
     initialDelaySeconds: 5
     timeoutSeconds: 5
    resources:
     limits:
      cpu: 2000m
      memory: 512Mi
     requests:
      cpu: 100m
      memory: 64Mi
```

Update an ACK orchestration template and use it to redeploy the application that is created from the template

If an application is created from an ACK orchestration template, you can run the following command to set a trigger to update the image of the demo Deployment and redeploy the application in the cluster.

```
curl -X POST -H 'content-type: application/json' https://cs.console.aliyun.com/hook/trigger?token=xxxxxxxxx -d '{"resource":{"deployment":{"metadata":{"name":"demo"}, "spec": {"template": {"spec": {"containers": [{"name":"demo","image":"registry.cn-hangzhou.aliyuncs.com/acs/rollouts-demo:red"}]}}}}}}'
```

> **Note**
> The body of the preceding command contains the JSON patch that is used to update the Deployment. The patch includes the following information about the Deployment: the name of the Deployment, the names of containers, and the address of the image registry.

```
{
  "resource":{
    "deployment":{
      "metadata":{
        "name":"demo"
      },
      "spec":{
        "template":{
          "spec":{
            "containers":[
              {
                "name":"demo",
                "image":"registry.cn-hangzhou.aliyuncs.com/acs/rollouts-demo:red"
              }]
          }
        }
      }
    }
  }
}
```

In this example, the trigger completes the following tasks:

- Updates the ACK orchestration template and automatically generates a new template version.
- Checks whether the data source of the application is updated. If the data source is updated, the latest ACK orchestration template is pulled and then used to redeploy the Deployment.
- Redeploys the application in the cluster by using the latest ACK orchestration template.

Redeploy an application that is created from a Git or Helm repository

If an application is created from a Git or Helm repository, you can run the following command to set a trigger to redeploy the application in the cluster. However, you cannot pass parameters to the trigger in the body of the command.

```
curl -X POST https://cs.console.aliyun.com/hook/trigger?token=xxxxxxxxx
```

In this example, the trigger completes the following tasks:

- Checks whether the data source of the application is updated. If the data source is updated, the latest template is pulled from the Git or Helm repository to redeploy the Deployment.
- Redeploys the application in the cluster by using the latest ACK orchestration template.

20.7. Deploy and manage an application in clusters across regions

If you have created multiple Container Service for Kubernetes (ACK) clusters across regions with your Alibaba Cloud account and the clusters use different environment variables, you can use Application Center to deploy an application to these clusters at the same time. This topic describes how to deploy an application to two ACK clusters that are deployed in different regions. The application is deployed by using two different Docker images.

Prerequisites

An ACK cluster with Internet access is created. For more information, see 创建Kubernetes托管版集群.

Step 1: Install Application Center

Deploy the Application Center controller in the primary cluster. The cluster named `ack-hangzhou` is the primary cluster where the Application Center controller is deployed in this example. A cluster named `ack-beijing` is added to Application Center as an external cluster. Then, you can manage both clusters in Application Center.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center (Previous Version)**.
3. On the **Applications** page, select `ack-hangzhou` on the **Install Controller** wizard page and click **Install**.
4. On the **Add Clusters** wizard page, find the cluster `ack-beijing` and click **Add** to add the cluster to Application Center.
5. Click **Get Started** to go to the Applications page.

Step 2: Create an application

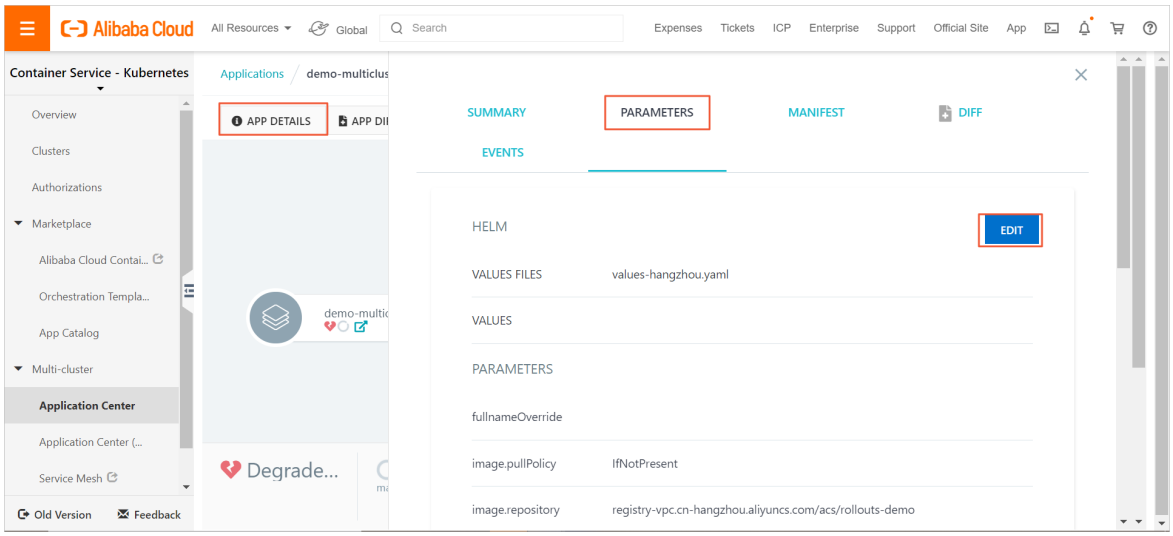
In this example, the application demo-multicluster is deployed in the `ack-hangzhou` and `ack-beijing` clusters that are in different regions. The application is deployed by using two different Docker images.

1. In the upper-right corner of the **Applications** page, click **+ NEW APP** and set the following parameters to create an application by using a Helm chart:
   - General settings
     - Application Name: demo-multicluster is entered in this example.
     - SYNC POLICY: Manual is selected in this example.
   - Source settings
     - Data source: GIT is selected in this example.
     - Repository URL: The repository URL is set to https://github.com/AliyunContainerService/appcenter-samples.git in this example.
     - Revision: The master branch is used in this example.
     - Path: The path is set to examples/demo-multicluster in this example.
   - Destination settings
     - Cluster: The cluster ack-hangzhou and the default namespace are used in this example.
     - Cluster: The cluster ack-beijing and the default namespace are used in this example.
   - Helm: The default file values.yaml is specified in this example.
2. After you set the parameters, click **CREATE**.
3. After you perform the preceding steps, two applications named **demo-cluster-0** and **demo-cluster-1** are displayed on the **Applications** page. Both of them belong to the same application group: **demo-multicluster**.

Step 3: Configure the applications based on their regions

Deploy the application **demo-cluster-0** in the cluster **ack-hangzhou**. Use the file *values-hangzhou.yaml* to set parameters. The **Docker image** used by the application is **registry-vpc.cn-hangzhou.aliyuncs.com/acs/rollouts-demo:blu** .

1. On the **Applications** page, click the **demo-cluster-0** card.
2. Click **APP DETAILS**. In the panel that appears, click the **PARAMETERS** tab and click **EDIT** .



3. Set **VALUES FILES** to **values-hangzhou.yaml** and click **SAVE**.

Deploy the application **demo-cluster-1** in the cluster **ack-beijing**. Use the *values-beijing.yaml* file to set parameters. The **Docker image** used by the application is **registry-vpc.cn-beijing.aliyuncs.com/acs/rollouts-demo:green** .

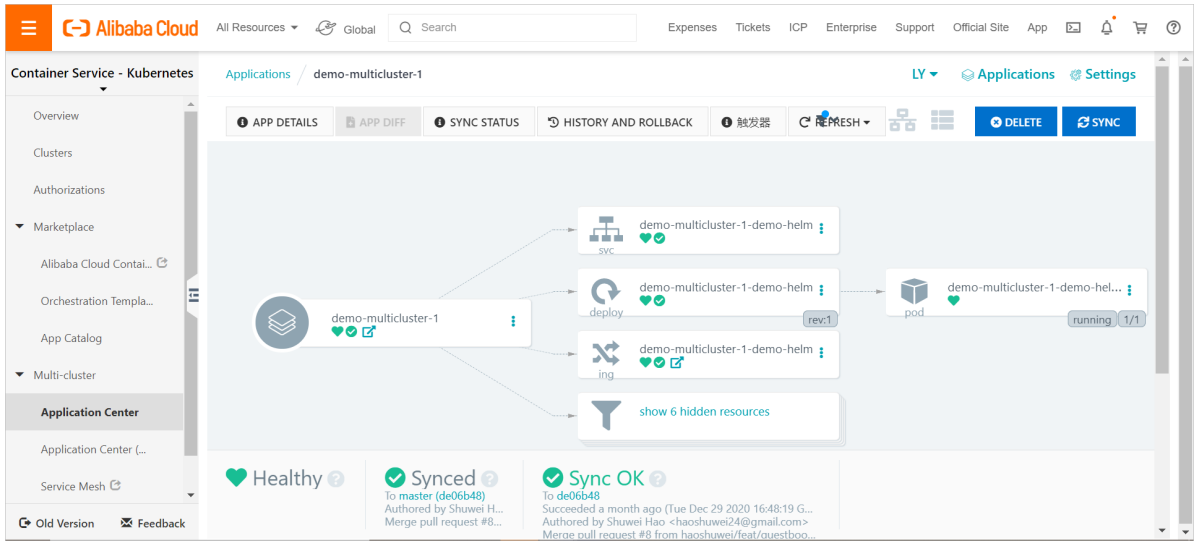
1. On the **Applications** page, click the **demo-cluster-1** card.
2. Click **APP DETAILS**. In the panel that appears, click the **PARAMETERS** tab and click **EDIT** .
3. Set **VALUES FILES** to **values-beijing.yaml** and click **SAVE**.

Step 4: Deploy the applications at a time

1. In the upper-right corner of the **Applications** page, click **SYNC APPS**
2. Select **APPLICATION SET: DEMO-MULTICLUSTER** and click **SYNC**.
3. If the **Complete** message appears, it indicates that the applications are deployed.

View the application topology and status

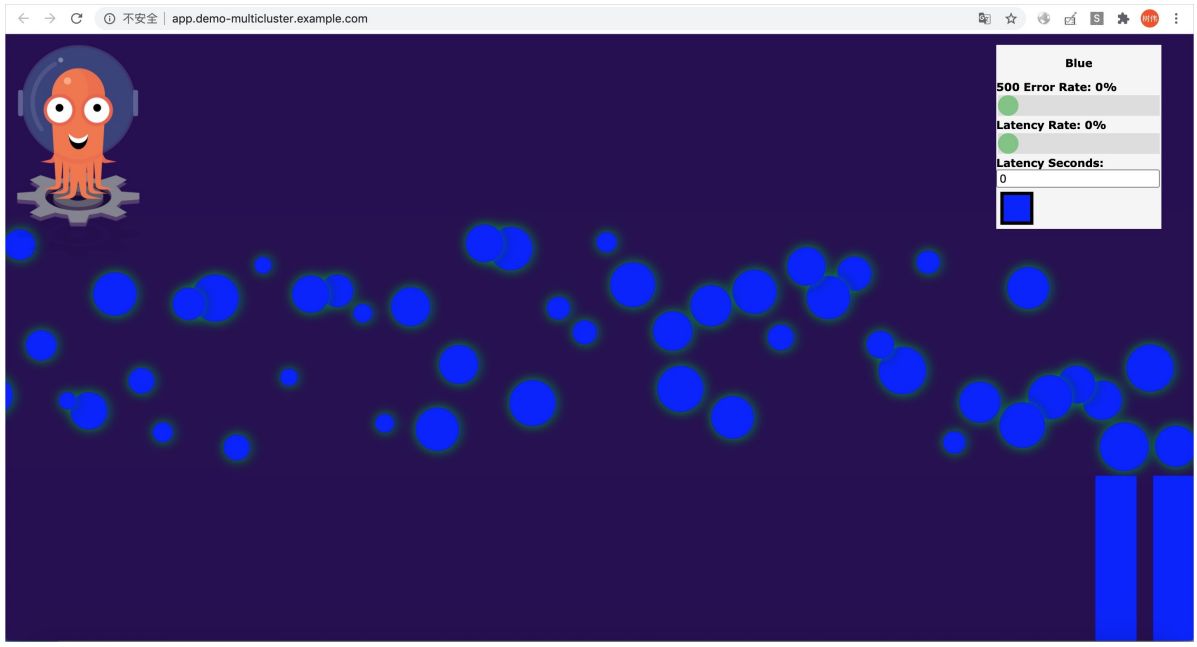
- On the **Applications** page, you can view the status of the applications that you have deployed.
- On the **Applications** page, click the card of the application that you want to view to go to the application details page. Then, you can view the topology and status of the application. The following figure shows the details page of the application **demo-multicluster-1**.



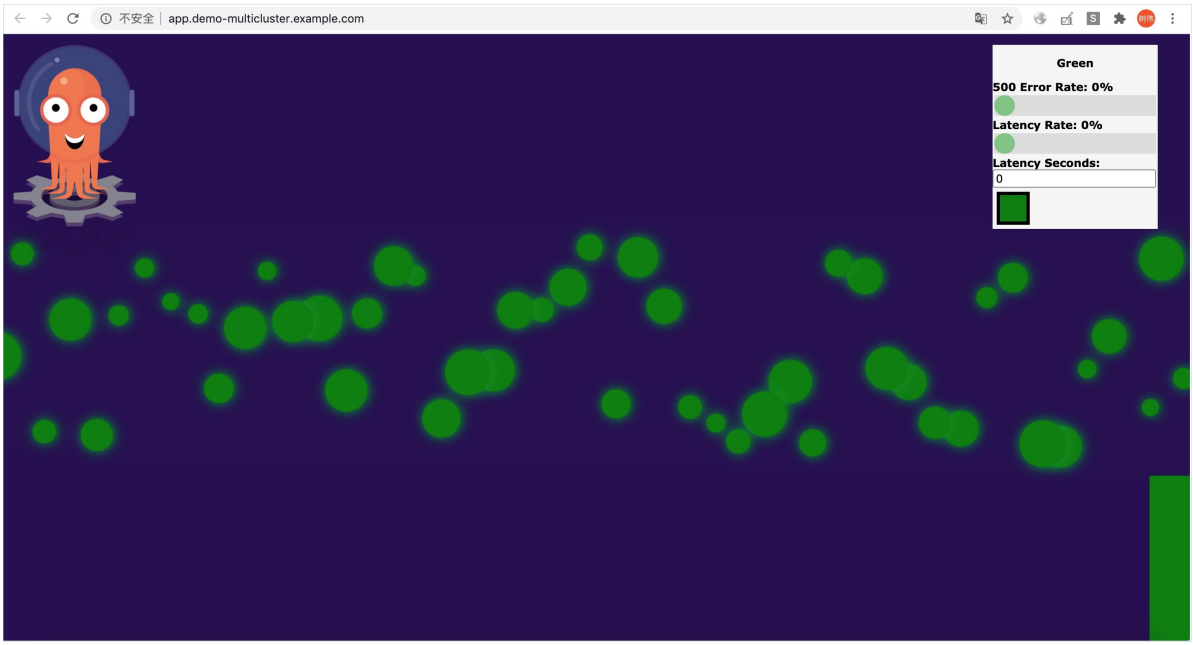
Access the applications

An NGINX Ingress is used to expose the application demo-multicluster in this example.

- Access the domain name of the application that is deployed in the cluster **ack-hangzhou**, as shown in the following figure.



- Access the domain name of the application that is deployed in the cluster **ack-beijing**, as shown in the following figure.



20.8. Create a trigger to automate application updates when the base image is updated

You can create triggers for each repository in Container Registry. After you create a trigger for a repository, a notification is sent to you when a container image in the repository is updated. When the base image of an application is updated and the update matches the trigger conditions, the updated base image is pulled to redeploy the application. This topic describes how to automate Application Center to pull an updated base image and use the image to redeploy the application by creating a trigger.

Prerequisites

A container image is pushed to a repository in Container Registry. For more information, see Use a Container Registry Enterprise Edition instance to push and pull images.

Step 1: Create an application

For more information, see Deploy an application in Application Center.

Step 2: Generate the webhook URL of a trigger in Application Center

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, choose **Multi-cluster > Application Center**.
3. On the **Applications** page, click the name of the application that you created.
4. On the application details page, click **Trigger** in the upper-right corner.
5. In the **Trigger** panel, click **Create**. After the webhook URL of the trigger appears, click **Copy**.

> **Note** We recommend that you keep the webhook URL confidential.

Step 3: Create a trigger in Container Registry

1. Log on to the Container Registry console.

2. In the top navigation bar, select a region.
3. In the left-side navigation pane, click **Instances**.
4. On the **Instances** page, click the required Container Registry Enterprise Edition instance.
5. On the management page of the Container Registry Enterprise Edition instance, , choose **Repositories > Repositories**.
6. On the **Repositories** page, click the name of the repository that is used when you created the application in Step 1.
7. On the repository details page, click **Trigger**. Then, click **Create**.
8. In the **Create Trigger** dialog box, specify the name of the trigger, enter the webhook URL of the trigger generated in Step 2, and set **Trigger** to **All**. Then, click **Confirm**.

   The following trigger modes are supported: **All**, **By RegExp**, and **By Tags**. The following section describes the trigger modes:

   - All: Each time an image is updated, Application Center pulls the updated image.
   - By RegExp: A regular expression is used to filter image updates. Application Center pulls an image only if the image is updated and the new image version matches the regular expression.
   - By Tags: Tags are used to filter image updates. Application Center pulls an image only if the image is updated and the new image version matches one of the specified tags.

   > **Note** If Application Center fails to pull an image, you can go to the **Trigger** page to find the trigger and click **Access Log** in the Actions column. In the **Access Log** message, check the cause of exceptions from the response bodies.

Step 4: Redeploy the application by using the updated container image

When a container image in the repository is updated, the updated image is automatically pulled and used to redeploy the application.

1. Run the following command to log on to the repository that is used to create the application in Step 1.

   > **Note** If the instance of Container Registry Enterprise Edition is created in the China (Hangzhou) region, you must set <the region where the instance of Enterprise Edition is created> to hangzhou in the command.

   ```
   docker login --username=<username used to log on to Container Registry> <name of the instance of Container Registry Enterprise Edition>-registry.cn-<the region where the instance of Container Registry Enterprise Edition is created>.cr.aliyuncs.com
   ```

2. Run the following command to add a tag to the image:

   ```
   docker tag registry.cn-hangzhou.aliyuncs.com/acs/rollouts-demo:green liusheng-registry.cn-beijing.cr.aliyuncs.com/liusheng/rollouts-demo:green
   ```

3. Run the following command to push the new version of the container image to the repository that is used when you created the application in Step 1.

   ```
   docker push liusheng-registry.cn-beijing.cr.aliyuncs.com/liusheng/rollouts-demo:green
   ```

Verify the result

1. View the orchestration template.

   After the container image is updated, Application Center pulls the updated image and updates the

orchestration template of the application.

i. Log on to the Container Service for Kubernetes (ACK) console.

ii. In the left-side navigation pane of the ACK console, choose **Marketplace > Orchestration Templates**.

iii. On the **Templates** page, find the template that you created in Step 1 and click **Details**. On the **Details** page, the updated image is used in the template.

2. Get the endpoint of the application.

i. Log on to the Container Service for Kubernetes (ACK) console.

ii. In the left-side navigation pane, click **Clusters**.

iii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

iv. In the left-side navigation pane of the details page, choose **Services and Ingresses > Ingresses**.

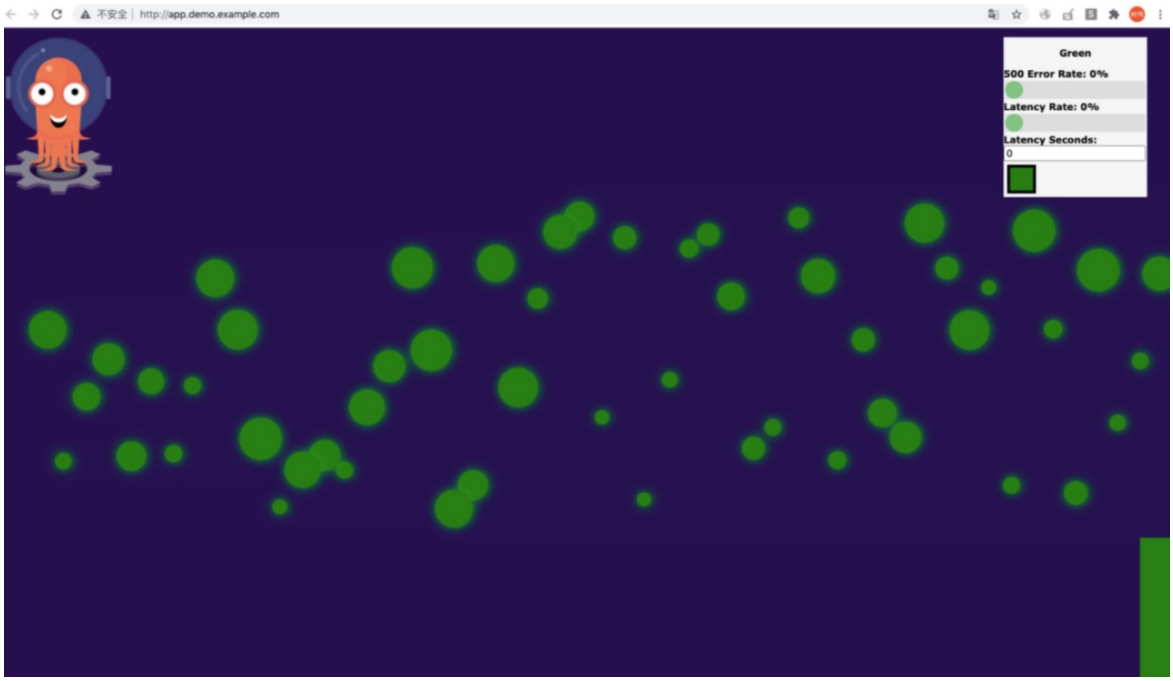
v. On the **Ingresses** page,view the endpoint of the demo.

3. Enter the following in the hosts file.

```
<endpoint> app.demo.example.com
```

4. Enter *app.demo.example.com* in the address bar of your browser.

If the page shown in the following figure appears in your browser, the application is updated.



21.Application marketplace

21.1. Container Registry

You can use Container Registry when you deploy applications. Container Registry can automatically build images and allows applications to automatically pull images. In addition, Container Registry improves the speed of image pulling and reduces the deployment time of applications.

Context

Container Registry Enterprise Edition is an enterprise-class platform designed to manage the lifecycle of cloud native application artifacts, including container images, Helm charts, and Open Container Initiative (OCI) artifacts. Container Registry Enterprise Edition efficiently distributes large-scale application artifacts across multiple regions in different scenarios. Container Registry Enterprise Edition seamlessly integrates with Container Service for Kubernetes (ACK), which simplifies the application delivery for enterprises.

For more information about how to get started with Container Registry, see Create a Container Registry Enterprise Edition instance and Use a Container Registry Enterprise Edition instance to push and pull images.

Features

- OCI artifact management: Container Registry Enterprise Edition can manage multiple types of OCI artifacts, such as container images for multiple architectures including Linux, Windows, and ARM, and charts of Helm v2 and Helm v3.
- Multi-dimensional security protection: Container Registry Enterprise Edition ensures storage and content security by storing encrypted cloud native application artifacts, scanning container images for vulnerabilities, and generating vulnerability reports in multiple dimensions. Container Registry Enterprise Edition ensures secure access by providing network access control and fine-grained operation audit for container images and Helm charts.
- Accelerated application distribution: Container Registry Enterprise Edition can synchronize container images among different regions around the world to improve the distribution efficiency. Container Registry Enterprise Edition supports P2P image distribution to accelerate application deployment and scaling.
- Efficient and secure cloud native application delivery: Container Registry Enterprise Edition allows you to create cloud native application delivery chains that are observable, traceable, and configurable. Based on delivery chains and blocking rules, Container Registry Enterprise Edition can automatically deliver applications all over the world upon source code changes in multiple scenarios. This improves the efficiency and security of cloud native application delivery.

Scenarios

- Global business deployment
  If your business is deployed around the world but you perform R&D and iteration activities in mainland China, you may not directly pull container images from regions in mainland China due to factors such as network instability. You can purchase Container Registry Enterprise Edition in global regions and configure rules to automatically synchronize container images. The automatic synchronization allows you to pull container images from the nearest region and implement cross-region disaster recovery when your business expands to different global regions.
- Large-scale business deployment
  If your business is large-scale and encounters bursts or iterations, you may not pull container images in time to cope with traffic peaks. Container Registry Enterprise Edition provides multiple instance types that allow you to simultaneously pull container images for all nodes in clusters with different scales. You can select an instance type based on the number of nodes in your cluster. Container Registry Enterprise Edition provides multiple methods to accelerate large-scale image distribution. This helps you ensure efficient container scaling and deployment.

- Efficient DevSecOps environment building based on containers
  If your business is deployed on platforms such as ACK, Alibaba Cloud Serverless Kubernetes, Enterprise Distributed Application Service (EDAS), and Elastic Container Instance (ECI), you must ensure security and efficiency of your business delivery chain. Container Registry Enterprise Edition allows you to create cloud native application delivery chains that are observable, traceable, and configurable. Based on the delivery chains, application artifacts can be automatically created, scanned for vulnerabilities, and signed upon source code changes. If risks are detected based on blocking rules, Container Registry Enterprise Edition automatically stops follow-up steps in the delivery chains. Application artifacts with no risks can be automatically distributed in different global regions. This allows you to deploy cloud native applications in different scenarios. Based on cloud native application delivery chains, you can include security into the business delivery process and upgrade DevOps to DevSecOps. In this way, you can ensure a more secure and efficient delivery of cloud native application artifacts.
- On-premises data migration to the cloud
  Migrating cloud native application artifacts to the cloud is the top priority in scenarios where you migrate on-premises data to the cloud. Container Registry Enterprise Edition allows you to efficiently import images from Object Storage Service (OSS) buckets to Container Registry Enterprise Edition instances. You can also use migration tools to smoothly migrate images from container image services of other service providers, such as Google, Azure, and AWS, to Container Registry Enterprise Edition.

21.2. Template management

21.2.1. Create an orchestration template

Orchestration templates define and describe a set of resources. You can use orchestration templates to create applications. This topic describes how to create an orchestration template.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > Orchestration Templates**.
3. On the **Templates** page, click **Create** in the upper-right corner.
4. In the **Create** dialog box, configure the template and click **Save**.
   - **Name**: Specify a name for the template.
   - **Description**: Specify a description for the template. This parameter is optional.
   - **Template**: Configure the template that conforms to the YAML syntax.

     > **Note**
     > - The template can contain multiple resource objects that are separated with `---`.
     > - You can add values in the format of `${params}` to the YAML file, for example, `app: ${nginx}`. The value is automatically identified as a parameter when you use the template to create an application.

   After the template is created, you are redirected to the **Templates** page. On the **My Templates** tab, you can find the created template.
5. (Optional)You can save a template provided by Container Service for Kubernetes (ACK) as an orchestration template.
   i. In the left-side navigation pane of the ACK console, click **Clusters**.

ii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.

iii. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

iv. On the **Deployments** page, click **Create from YAML** in the upper-right corner.

v. Select a template and click **Save Template**.

vi. In the **Save Template** dialog box, specify the name, description, and content of the template. Then, click **Save**.

> **Note** You can modify the template based on your requirements.

What's next

On the **My Templates** tab, you can specify a template to create an application.

21.2.2. Modify an orchestration template

This topic describes how to modify an orchestration template.

Prerequisites

An orchestration template is created. For more information, see Create an orchestration template.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > Orchestration Templates**.
3. Find the template that you want to modify and click **Details**.
4. On the template details page, set the **Cluster**, **Namespace**, and **Version Description** parameters and modify the configurations in the template based on your requirements. Then, click **Save** in the Deploy section.
5. Go to the **Templates** page. On the **My Templates** tab, you can find that the template is updated.

21.2.3. Save an orchestration template as a new one

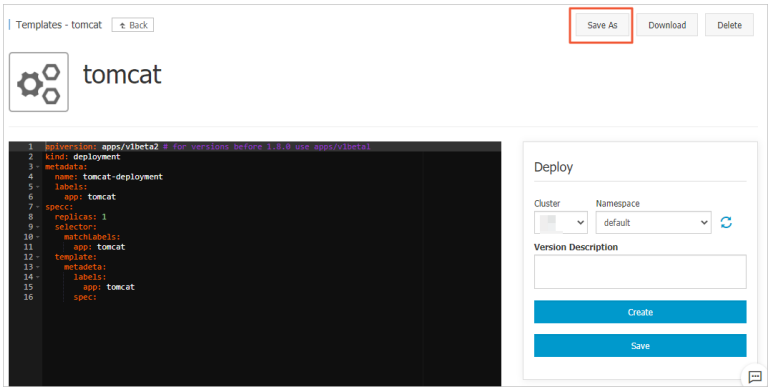
This topic describes how to save an existing orchestration template as a new one in the Container Service for Kubernetes (ACK) console.

Prerequisites

An orchestration template is created. For more information, see Create an orchestration template.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > Orchestration Templates**.
3. Find the template that you want to modify and click **Details**.
4. On the template details page, modify the parameters and the template based on your requirements and click **Save As** in the upper-right corner of the page.



5. In the **Save Template As** dialog box, specify the name of the template and then click **OK**.
   Go to the **Templates** page. On the **My Templates** tab, you can find the new template.

21.2.4. Download an orchestration template

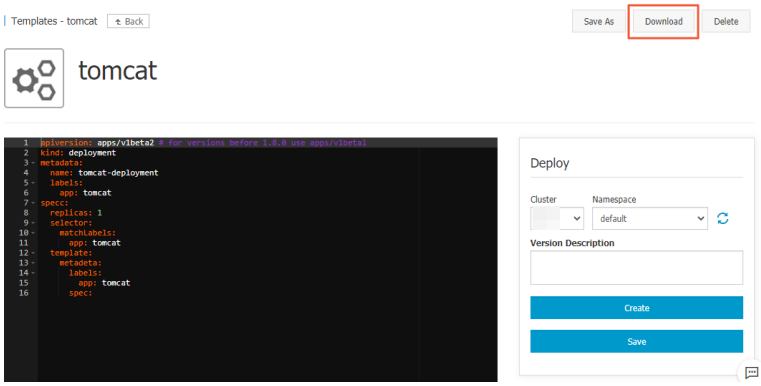
This topic describes how to download an orchestration template in the Container Service for Kubernetes (ACK) console.

Prerequisites

An orchestration template is created. For more information, see Create an orchestration template.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > Orchestration Templates**.
3. Find the template that you want to modify and click **Details**.
4. In the upper-right corner of the template details page, click **Download**. The template is saved to a .yml file.



21.2.5. Delete an orchestration template

This topic describes how to delete an orchestration template in the Container Service for Kubernetes (ACK) console.

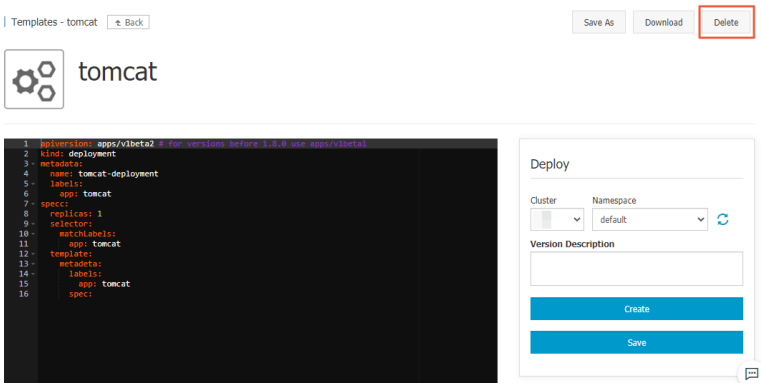
Prerequisites

An orchestration template is created. For more information, see Create an orchestration template.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.

2. In the left-side navigation pane of the ACK console, choose **Marketplace > Orchestration Templates**.
3. Find the template that you want to modify and click **Details**.
4. In the upper-right corner of the template details page, click **Delete**.



5. In the **Note** message, click **OK**.

21.3. App catalog management

21.3.1. Overview

App Catalog of Container Service for Kubernetes (ACK) is integrated with Helm. App Catalog allows you to use features provided by Helm and also supports new features such as graphical user interfaces and Alibaba Cloud repositories.
Microservice is the main component of containerization. However, it is challenging to deploy and manage applications as microservices. An application can be divided into several microservices. These microservices can be separately deployed and expanded to achieve agile development and fast iteration. Microservices have many benefits. However, a large number of microservices pose challenges to developers who need to manage microservices, including resources, versions, and configurations.

ACK integrates Helm to simplify how you can deploy and manage Kubernetes applications as microservices on the Kubernetes orchestration platform.

Helm is an open source subproject for Kubernetes service orchestration. It functions as a package manager, which is used to package applications into versioned archives. Packaged Kubernetes applications are much easier to deploy and manage.

Features of App Catalog

The chart list on the App Catalog page provides the following information:

- Chart name: Each Helm chart corresponds to an application. The chart contains the image, dependencies, and resource definitions that are required to run the application.
- Version: The version of a chart.
- Repository: The repository that is used to publish and store charts. Repositories provided by Helm are stable and incubator.

The information displayed on the details page of each chart varies. It may contain the following items:

- Chart introduction
- Chart details
- Prerequisites for installing a chart to an ACK cluster. For example, you must first configure a persistent volume (PV).
- Chart installation commands

- Chart uninstallation commands
- Chart parameters

You can use Helm to deploy and manage charts in App Catalog. For more information, see Use Helm to simplify application deployment.

Disclaimer

Some programs in App Catalog are developed from open source programs to better fit ACK. Alibaba Cloud provides comprehensive technical support for these programs in App Catalog. However, Alibaba Cloud is not responsible for any compensation, reimbursement, or damages arising in connection with the defects of the open source programs that you use.

21.3.2. View the application catalog

You can view the application catalog in the Container Service for Kubernetes (ACK) console.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.

   On the App Catalog page, click the **Alibaba Cloud Apps** or the **App Hub** tab. On both tabs, you can click and view information about each chart, including basic information, application name, image version, and image repository. The chart information can be used to deploy an application.

What's next

You can click a chart and obtain the detailed information about the chart. You can deploy the application based on the related information by using the Helm tool.

22.Virtual nodes and ECI

22.1. Deploy the virtual node controller and use it to create Elastic Container Instance-based pods

Virtual nodes enable seamless integration between Kubernetes and Elastic Container Instance. Virtual nodes empower Kubernetes clusters with high elasticity. This way, Kubernetes clusters are no longer limited by the computing capacity of cluster nodes. You can dynamically create Elastic Container Instance-based pods to meet your business requirements. This saves the trouble of cluster sizing. This topic describes virtual nodes and elastic container instances. It also describes how to deploy the virtual node controller (ack-virtual-node) and then use it to create Elastic Container Instance-based pods.

Prerequisites

- A managed Kubernetes cluster or a dedicated Kubernetes cluster with a version later than V1.14 is created. For more information, see 创建Kubernetes托管版集群 or Create a dedicated Kubernetes cluster.
- Elastic Container Instance is activated. You can log on to the Elastic Container Instance console to activate the service.
- The region where the cluster is deployed must be supported by elastic container instances. To view the supported regions and zones, log on to the Elastic Container Instance console.

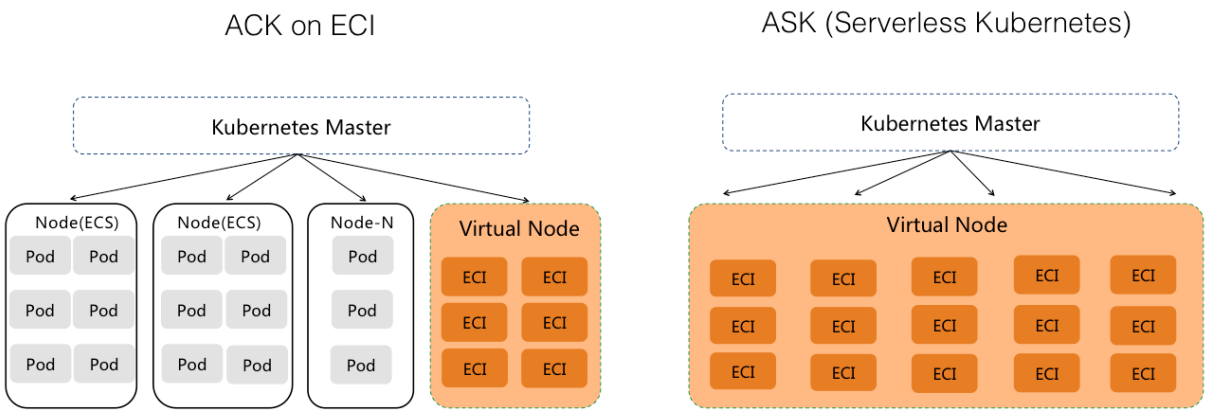
Virtual nodes and elastic container instances

Elastic Container Instance is a serverless compute service that is provided by Alibaba Cloud for containerization. You can use elastic container instances to set up an operations and maintenance (O&M)-free and isolated runtime environment for your containers. Elastic container instances allow you to focus on containerized applications without the need to purchase or manage Elastic Compute Service (ECS) instances. This way, you do not need to perform infrastructure maintenance. You can create elastic container instances to meet your business requirements. You are charged for resource usage on a per second basis.

Virtual nodes enable seamless integration between Kubernetes and Elastic Container Instance. Virtual nodes empower Kubernetes clusters with high elasticity. This way, Kubernetes clusters are no longer limited by the computing capacity of cluster nodes. You can dynamically create Elastic Container Instance-based pods to meet your business requirements. This saves the trouble of cluster sizing. Virtual nodes can significantly reduce computing costs and improve cluster elasticity in the following scenarios:

- Online business that requires elastic scaling to withstand traffic fluctuations, such as online education and e-commerce. Virtual nodes optimize the maintenance of resource pools. This can help you reduce computing costs.
- Virtual nodes can reduce costs in computing scenarios where Spark or Presto is used to process data.
- CI/CD pipeline: Jenkins and Gitlab-Runner.
- Jobs: Jobs in Artificial Intelligence (AI) computing scenarios and CronJobs.

Based on virtual nodes and elastic container instances, ACK provides multiple serverless container services, such as serverless Kubernetes (ASK) and ACK on Elastic Container Instance. You can use these services to deploy elastic and maintenance-free workloads.



Step 1: Deploy ack-virtual-node in ACK clusters

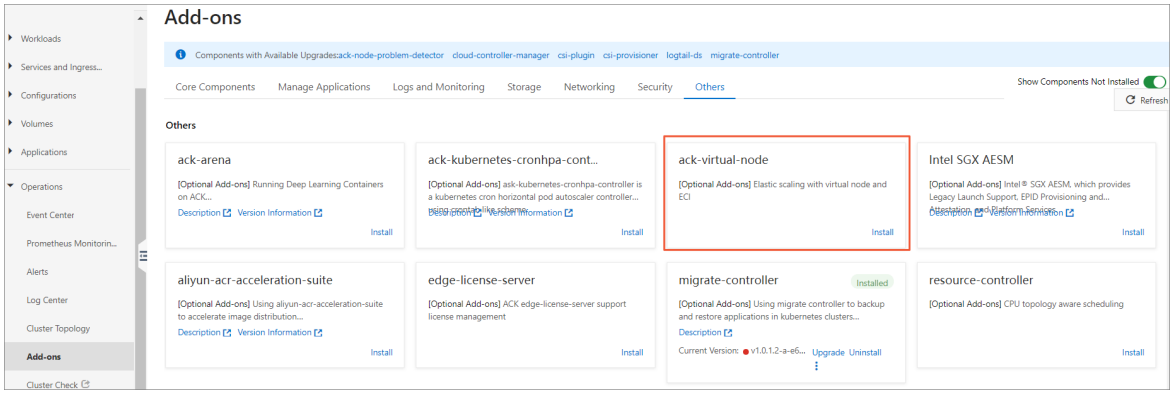
> **Note**
>
> - The virtual node controller ack-virtual-node is managed by ASK clusters. Therefore, you can create Elastic Container Instance-based pods in ASK clusters without the need to deploy ack-virtual-node. In managed or dedicated Kubernetes clusters, you must deploy ack-virtual-node before you can create Elastic Container Instance-based pods.
> - In managed Kubernetes clusters, you can deploy ack-virtual-node on the **Add-ons** page. By default, ack-virtual-node is managed by the clusters after it is deployed.
> - In dedicated Kubernetes clusters, you can deploy ack-virtual-node on the **App Catalog** page. By default, ack-virtual-node is managed by the clusters after it is deployed.

Managed Kubernetes clusters

Perform the following steps to deploy ack-virtual-node in a managed Kubernetes cluster:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. On the **Add-ons** page, select **ack-virtual-node** and click **Install**.

   The default vSwitch and security group of the cluster are used for elastic container instances that are deployed by ack-virtual-node. If you want to modify these settings, see What to do next.



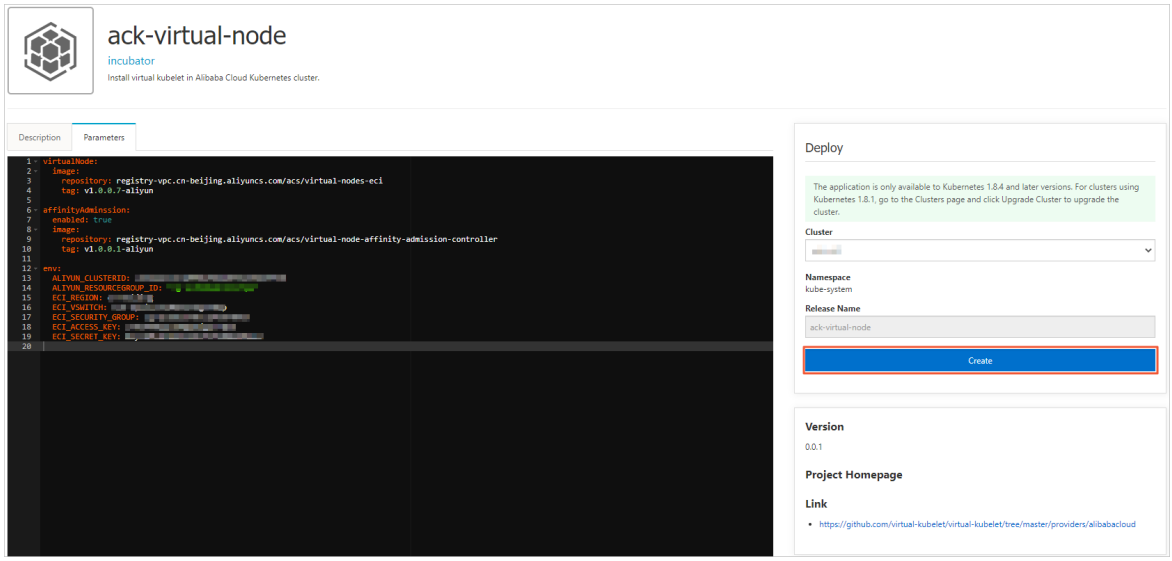
Dedicated Kubernetes Clusters

Perform the following steps to deploy ack-virtual-node in a dedicated Kubernetes cluster:

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab and find and click **ack-virtual-node**.

   In the upper-right corner of the **App Catalog** page, you can enter **ack-virtual-node** into the Name search bar and click the search icon. You can also enter a keyword to perform a fuzzy match.
4. On the **App Catalog - ack-virtual-node** page, select a cluster in the **Deploy** section.

   Namespace is automatically set to *kube-system* and **Release Name** is automatically set to *ack-virtual-node*.
5. On the **App Catalog - ack-virtual-node** page, click the **Parameters** tab and set the parameters. Then, click **Create** in the **Deploy** section.



Parameter	Description	How to obtain the value
ALIYUN_CLUSTERID	The ID of the cluster where you want to deploy ack-virtual-node	On the details page of the cluster, click the **Basic Information** tab. In the **Basic Information** section, you can view the **ID of the cluster**.

Parameter	Description	How to obtain the value
ALIYUN_RESOURCE GROUP_ID	The ID of the resource group to which the cluster belongs	If you do not specify the parameter, the default resource group is used. To specify a resource group, log on to the Resource Management console to obtain the **ID of the resource group**.
ECI_REGION	The name of the region where the cluster is deployed	On the details page of the cluster, click the **Basic Information** tab. In the **Basic Information** section, you can view the **Region** where the cluster is deployed. **Note** For example, cn-hangzhou indicates the China (Hangzhou) region.
ECI_VSWITCH	vSwitches	On the **Nodes** page, click the ID of a node. On the **Instance Details** page, you can obtain the value of VSwtich in the **Configuration Information** section. **Note** Make sure that the zone of the vSwitch is supported by elastic compute instances. You can specify vSwitches that are deployed in different zones. For example, you can specify multiple vSwitches in the format of `ECI_VSWITCH: "vsw-xxxxxxx1, vsw-xxxxxxx2, vsw-xxxxxxx3"`.
ECI_SECURITY_GROUP	The ID of the security group to which nodes in the cluster belong	On the **Nodes** page, click the ID of a node. In the left-side navigation pane, click **Security Groups**. Click the **Security Groups** tab to obtain the ID of the security group.
ECI_ACCESS_KEY	The AccessKey ID of your Alibaba Cloud account	For more information, see Obtain an AccessKey pair. You must attach the **AliyunECIReadOnlyAccess** policy to your account in the RAM console. For more information, see Grant permissions to a RAM user.
ECI_SECRET_KEY	The AccessKey secret of your Alibaba Cloud account	For more information, see Obtain an AccessKey pair. You must attach the **AliyunECIReadOnlyAccess** policy to your account in the RAM console. For more information, see Grant permissions to a RAM user.

6. Run the following command to view the status of **virtual-node-controller**. For more information, see Use kubectl on Cloud Shell to manage ACK clusters.

   Run the following command:

   ```
   kubectl -n kube-system get deploy ack-virtual-node-controller
   ```

   The following output is returned:

   ```
   NAME                       READY  AGE
   virtual-node-controller 1/1    1m
   ```

Step 2: Create Elastic Container Instance-based pods

> **Note** All the pods in ASK clusters are Elastic Container Instance-based pods. You do not need to use labels to identify Elastic Container Instance-based pods. Therefore, the following operations do not apply to ASK clusters. The following operations show how to create an Elastic Container Instance-based pod in an ACK cluster.

You can use one of the following methods to create an Elastic Container Instance-based pod in an ACK cluster.

- **Add a label to the pod.**
  Add the alibabacloud.com/eci=true label to the pod. The pod becomes an Elastic Container Instance-based pod and is scheduled to the virtual node. Example:
  Run the following command:

  ```
  kubectl run nginx --image nginx -l alibabacloud.com/eci=true
  ```

  ```
  kubectl get pod -o wide|grep virtual-kubelet
  ```

  The following output is returned:

  ```
  nginx-7fc9f746b6-r4xgx   0/1   ContainerCreating  0     20s  192.168.*.*  virtual-kubelet    <none>     <none
  >
  ```

- **Add a label to the namespace where the pod is deployed.**
  Add the alibabacloud.com/eci=true label to the namespace where the pod is deployed. The pod becomes an Elastic Container Instance-based pod and is scheduled to the virtual node. Example:
  Run the following command:

  ```
  kubectl create ns vk
  ```

  ```
  kubectl label namespace vk alibabacloud.com/eci=true
  ```

  ```
  kubectl -n vk run nginx --image nginx
  ```

  ```
  kubectl -n vk get pod -o wide|grep virtual-kubelet
  ```

  The following output is returned:

  ```
  nginx-6f489b847d-vgj4d   1/1   Running      0     1m  192.168.*.*  virtual-kubelet    <none>     <none>
  ```

What to do next

> **Note** The following operations only apply to ACK clusters and do not apply to ASK clusters.

Delete virtual nodes in ACK clusters

1. Uninstall ack-virtual-node.
   - After you delete all the pods in a managed Kubernetes cluster, you can uninstall ack-virtual-node on the **Add-ons** page.
   - After you delete all the Elastic Container Instance-based pods in a dedicated Kubernetes cluster, you can uninstall ack-virtual-node on the **Helm** page.
2. Run the `kubectl delete no` command to delete related virtual nodes.

> **Note** If you do not delete the Elastic Container Instance-based pods in the cluster before you uninstall ack-virtual-node, the elastic container instances are retained in the cluster.

Related information

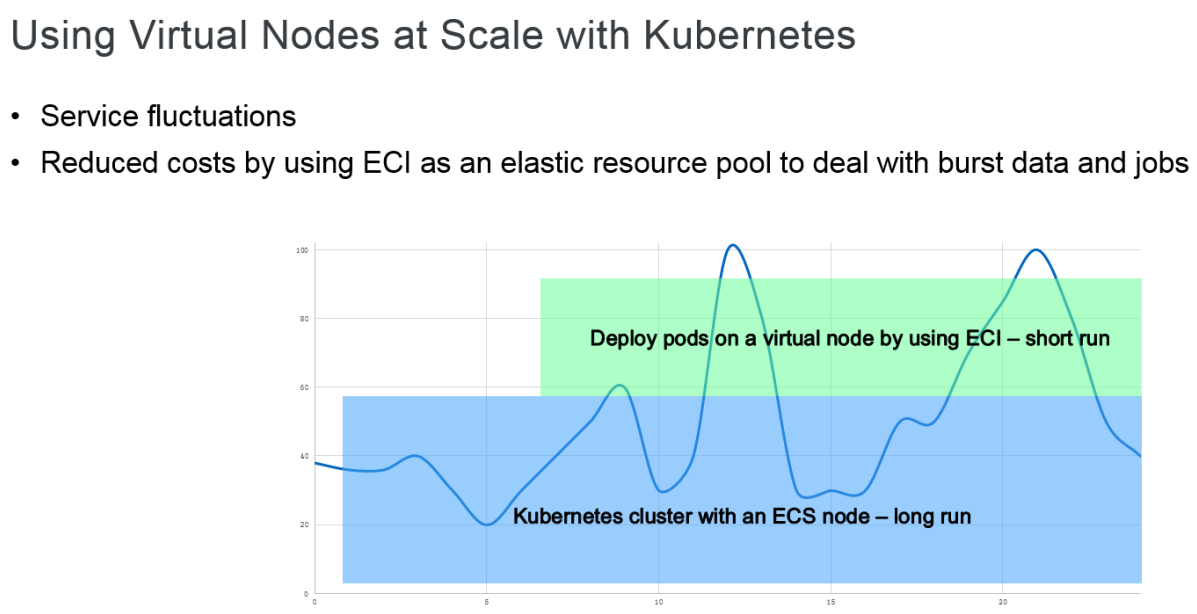
- Run a job by using a virtual node
- Use elastic container instances in ASK clusters

22.2. Run a job by using a virtual node

This topic describes how to run a job by using a virtual node. This method minimizes cluster O&M costs because you do not need to create new nodes in your cluster to handle peak load.

Context

Pods cannot run as expected when the nodes in a cluster have insufficient computing resources. However, creating a large number of nodes in a cluster may result in resource wastes. A virtual node offers near unlimited computing resources to a Kubernetes cluster and avoids resource wastes such as idle nodes. It allows you to create pods in a cluster based on your business requirements and ensures that the cluster can handle fluctuations in resource demands.



For example, you create a managed Kubernetes cluster. The cluster contains two worker nodes and a managed master node that is free of change. The specification of each worker node is 4 cores and 8 GB of memory. Therefore, the total computing capacity of the cluster is 8 cores and 16 GB of memory. You want to run an offline job to process data. This job requires 16 cores and 32 GB of memory. You cannot run the job in the managed Kubernetes cluster because the cluster can provide only 8 cores and 16 GB of memory. The computing resources provided by the cluster do not meet the requirement of the job. To resolve this problem, you can schedule the job to a virtual node and run the job without using the computing resources of the nodes in the cluster.

Prerequisites

- A managed Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群.
- A virtual node is deployed in the cluster. For more information, see Deploy the virtual node controller and use it to create Elastic Container Instance-based pods.
- The *virtual-node-affinity-injection: enabled* label is added to the namespace vk. For more information, see

create a pod by adding a label to a namespace.

Procedure

1. Connect to Kubernetes clusters by using kubectl.
2. Copy the following content to *job.yaml*, and run the `kubectl -n vk apply -f job.yaml` command to deploy a job in the cluster:

```
kind: Job
metadata:
 name: pi
spec:
 template:
  spec:
   containers:
   - name: pi
    image: perl
    command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
    resources:
     requests:
      cpu: 16
      memory: 32Gi
   restartPolicy: Never
 backoffLimit: 4
```

3. To query the state of the pod,

   run the following command:

```
kubectl -n vk get pod -a
```

   The following response is returned:

```
NAME    READY   STATUS    RESTARTS  AGE
pi-7cmwv  0/1    Completed  0      2m
```

   Run the following command:

```
kubectl -n vk describe pod
```

   The following response is returned:

```
Name:          pi-7cmwv
Namespace:      vk
Priority:       0
PriorityClassName: <none>
Node:          virtual-kubelet/
...
Events:
 Type   Reason          Age  From         Message
 ----   ------          ---- ----         -------
 Normal Scheduled       3m   default-scheduler Successfully assigned vk/pi-7cmwv to virtual-kubelet
 Normal SuccessfulMountVolume 2m   kubelet, eci     MountVolume.SetUp succeeded for volume "default-t
oken-b2tff"
 Normal Pulling         2m   kubelet, eci     pulling image "perl"
 Normal Pulled          2m   kubelet, eci     Successfully pulled image "perl"
 Normal Created          2m   kubelet, eci     Created container
 Normal Started          2m   kubelet, eci     Started container
```

> **Note** The pod on the virtual node is charged based on the amount of resources that you use. The system stops billing when the pod completes the job.

Running a job on a virtual node can reduce computing costs and O&M workloads. In addition, you do not need to worry about whether the cluster has sufficient computing resources or whether you need to scale in or out the number of nodes.

22.3. Deploy applications that provide services by using Ingresses

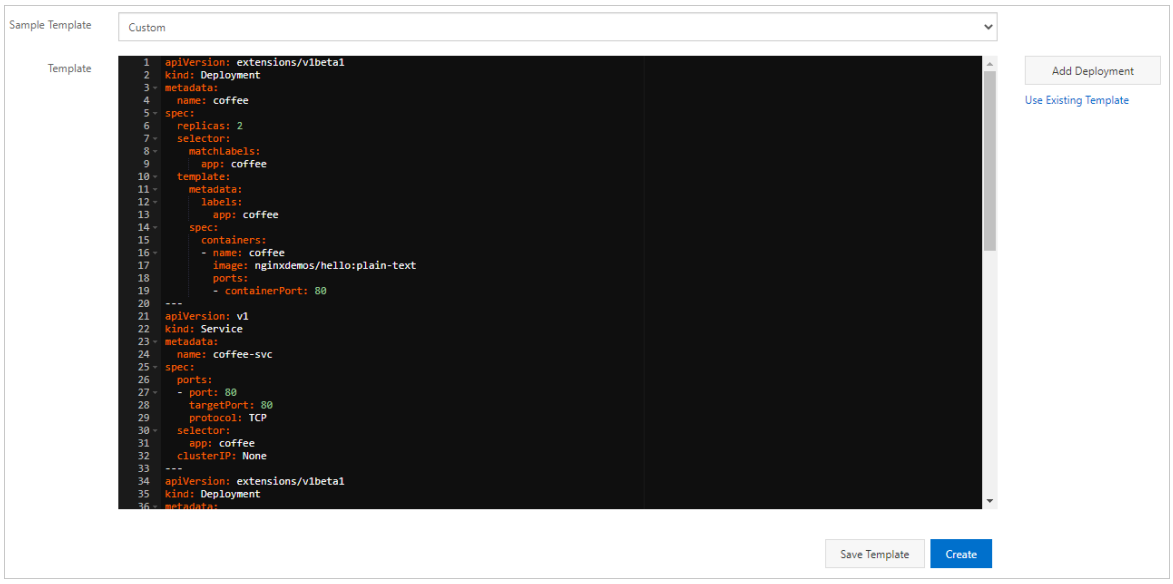
This topic describes how to deploy applications that provide services by using an Ingress on a virtual node of a Container Service for Kubernetes (ACK) cluster. This allows you to provide the applications with scalable and unlimited computing capacities without the need to create new nodes in the cluster. This also ensures the elasticity of the applications to withstand traffic fluctuations.

Prerequisites

- A virtual node is deployed in your ACK cluster. For more information, see Deploy the virtual node controller and use it to create Elastic Container Instance-based pods.
- The *virtual-node-affinity-injection: enabled* label is added to the vk namespace. For more information, see Create a pod in a namespace with specified labels.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. In the upper-right corner of the page, click **Create from Template**.
6. Select a sample template or customize a template, and click **Create**.



You can use the following YAML template to create applications and an Ingress that is used to enable access to the applications:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: coffee
spec:
 replicas: 2
 selector:
  matchLabels:
   app: coffee
 template:
  metadata:
   labels:
    app: coffee
  spec:
   containers:
   - name: coffee
    image: nginxdemos/hello:plain-text
    ports:
    - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
 name: coffee-svc
spec:
 ports:
 - port: 80
  targetPort: 80
  protocol: TCP
 selector:
  app: coffee
 clusterIP: None
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: tea
spec:
 replicas: 3
 selector:
  matchLabels:
   app: tea
 template:
  metadata:
   labels:
    app: tea
  spec:
   containers:
   - name: tea
    image: nginxdemos/hello:plain-text
    ports:
    - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
 name: tea-svc
```

```
  name: tea-svc
  labels:
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  selector:
    app: tea
  clusterIP: None
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

Verify the result

- In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**. You can find the newly created coffee and tea applications.
- In the left-side navigation pane of the cluster details page, choose **Workloads > Pods**. You can verify that the pods of the newly created applications run on virtual-kubelet nodes.
- On the details page of the cluster, choose **Network > Ingresses**. You can find the newly created Ingress.
- Run the following command to query the Ingress. Then, test access to the Ingress.

  ```
  kubectl get ing
  ```

  Expected output:

  ```
  NAME      HOSTS         ADDRESS      PORTS  AGE
  cafe-ingress  cafe.example.com  114.55.252.185  80    6m18s
  ```

  Run the following command to access `"Host:cafe.example.com" <EXTERNAL_IP>/tea` to test whether the tea application can be accessed:

  ```
  curl -H "Host:cafe.example.com" <EXTERNAL_IP>/tea
  ```

  Expected output:

  ```
  Server address: 192.168.xx.xx:80
  Server name: tea-658d56f6cc-cxxxx
  Date: 25/Sep/2020:12:36:50 +0000
  URI: /tea
  Request ID: b01d5bab9ae07abb8bc385377193xxxx
  ```

Run the following command to access `"Host:cafe.example.com" <EXTERNAL_IP>/coffee` to test whether the coffee application can be accessed:

```
curl -H "Host:cafe.example.com" <EXTERNAL_IP>/coffee
```

Expected output:

```
Server address: 192.168.xx.xx:80
Server name: coffee-8c8ff9b4f-hxxxx
Date: 25/Sep/2020:12:36:47 +0000
URI: /coffee
Request ID: 722fe41a65a7fb552613c56e0a9axxxx
```

22.4. Use ECI elastic scheduling

Elastic Container Instance (ECI) elastic scheduling is an elastic scheduling strategy provided by Alibaba Cloud. You can add annotations to specify the resources that you want to use when you deploy applications. You can use only Elastic Compute Service (ECS) instances or elastic container instances, or automatically request elastic container instances when ECS resources are insufficient. ECI elastic scheduling can meet your resource requirements in different workload scenarios.

For more information, see Use ECI elastic scheduling.

22.5. Enable a virtual node to discover Services by using Alibaba Cloud DNS PrivateZone

You can enable virtual nodes of Container Service for Kubernetes (ACK) to discover the following Services: intranet Services, headless Services, and ClusterIP Services.

Prerequisites

- Alibaba Cloud DNS PrivateZone is activated. You can activate the service in the Alibaba Cloud DNS console.
- A virtual node is created in an ACK cluster. For more information, see Deploy the virtual node controller and use it to create Elastic Container Instance-based pods.
- You have connected to the ACK cluster. For more information, see Connect to Kubernetes clusters by using kubectl.

Prerequisites

After you use a Helm chart to deploy a virtual node, verify that the relevant environment variables are valid.

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **Alibaba Cloud Apps** tab, click **ack-virtual-node**.
4. In the left-side navigation pane, choose **Marketplace > App Catalog** and then click **ack-virtual-node**.
5. On the **App Catalog - ack-virtual-node** page, click the **Parameters** tab to modify the parameters, as shown in the following figure.

```
virtualNode:
  image:
    repository: registry.cn-hangzhou.aliyuncs.com/acs/virtual-nodes-eci
    tag: v1.0.0.1-aliyun

affinityAdminssion:
  enabled: true
  image:
    repository: registry.cn-hangzhou.aliyuncs.com/ask/virtual-node-affinity-admission-controller
    tag: latest

env:
  ECI_REGION: "cn-hangzhou"
  ECI_VPC: "vpc-
  ECI_VSWITCH: "vsw-
  ECI_SECURITY_GROUP:
  ECI_ACCESS_KEY:
  ECI_SECRET_KEY:
  ALIYUN_CLUSTERID:
```

- `ECI_VPC`: The ID of the virtual private cloud (VPC) where the ACK cluster is deployed. Replace it with the VPC ID of the current cluster.
- `ALIYUN_CLUSTERID`: The ID of the ACK cluster. Replace it with the ID of the current cluster. You are not allowed to set the value to default or leave this parameter empty.

Procedure

1. Create a Deployment and Services.

   You can add the following sample code to a YAML file and run the `kubectl create -f nginx-service-ack.yaml` command to create a Deployment and Services:

```
apiVersion: v1
kind: Service
metadata:
 name: nginx-headless-service
 annotations:
   service.beta.kubernetes.io/alibaba-cloud-private-zone-enable: "true"
spec:
 ports:
 - port: 80
   protocol: TCP
 selector:
   app: nginx
 clusterIP: None
---
apiVersion: v1
kind: Service
metadata:
 name: nginx-clusterip-service
 annotations:
   service.beta.kubernetes.io/alibaba-cloud-private-zone-enable: "true"
spec:
 ports:
 - port: 80
   protocol: TCP
 selector:
   app: nginx
 type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
 name: nginx-intranet-service
 annotations:
   service.beta.kubernetes.io/alicloud-loadbalancer-address-type: intranet
```

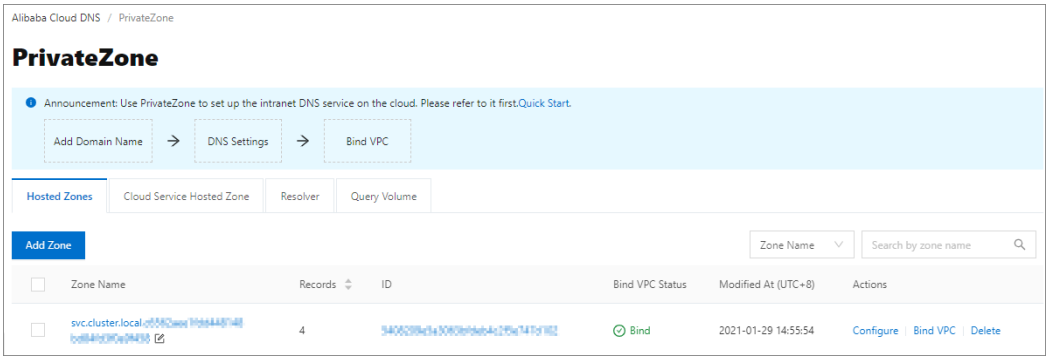
```
    service.beta.kubernetes.io/alibaba-cloud-private-zone-enable: "true"
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    app: nginx
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image:  nginx:alpine
        ports:
        - containerPort: 80
```

Note By default, the DNS records of Services in an ACK cluster are not synchronized to Alibaba Cloud DNS PrivateZone. To synchronize these records, you must add the following annotation to the Service YAML file:

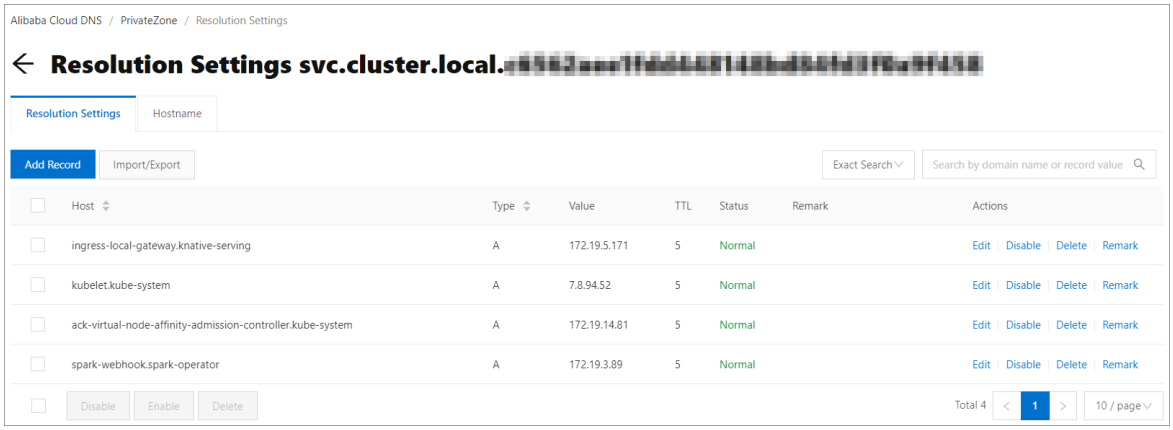
```
annotations:
  service.beta.kubernetes.io/alibaba-cloud-private-zone-enable: "true"
```

This way, the virtual node Controller synchronizes the DNS records of the Services to Alibaba Cloud DNS PrivateZone.

2. Log on to the Alibaba Cloud DNS console.
3. In the left-side navigation pane, click **PrivateZone**. On the **Hosted Zones** tab, you can view the PrivateZone records that are automatically generated in each zone.



4. In the list of Hosted Zones, find the zone that you want to configure and click **Configure** in the Actions column. The Resolution Settings page appears.



> **Note** All PrivateZone records are in the `$svc.$ns` format. Each PrivateZone record maps a Service to an IP address. The following resolution rules apply:
>
> - A LoadBalancer Service corresponds to only one PrivateZone record. The record maps the Service to the IP address of the Server Load Balancer (SLB) instance.
> - A ClusterIP Service corresponds to multiple PrivateZone records. The records map the Service to the IP addresses of backend pods.
> - A headless Service corresponds to multiple PrivateZone records. The records map the Service to the IP addresses of backend pods.

You can access a Service within the VPC through the private domain name of the Service.

- You can use `$svc.$ns.svc.cluster.local.$clusterId` to access Services that are discovered in the current cluster. You can also use the long domain name to access Services in other clusters after you use Alibaba Cloud DNS PrivateZone to discover these Services.
- You can use `$svc` to access Services in the current namespace and use `$svc.$ns` to access Services in other namespaces.

22.6. Install virtual-kubelet-autoscaler

Alibaba Cloud provides the virtual-kubelet-autoscaler plug-in for Container Service for Kubernetes (ACK) clusters. If a pod fails to be scheduled to Elastic Compute Service (ECS) nodes in a cluster (for example, due to insufficient node resources), virtual-kubelet-autoscaler reschedules the pod to a virtual node deployed on an elastic container instance.

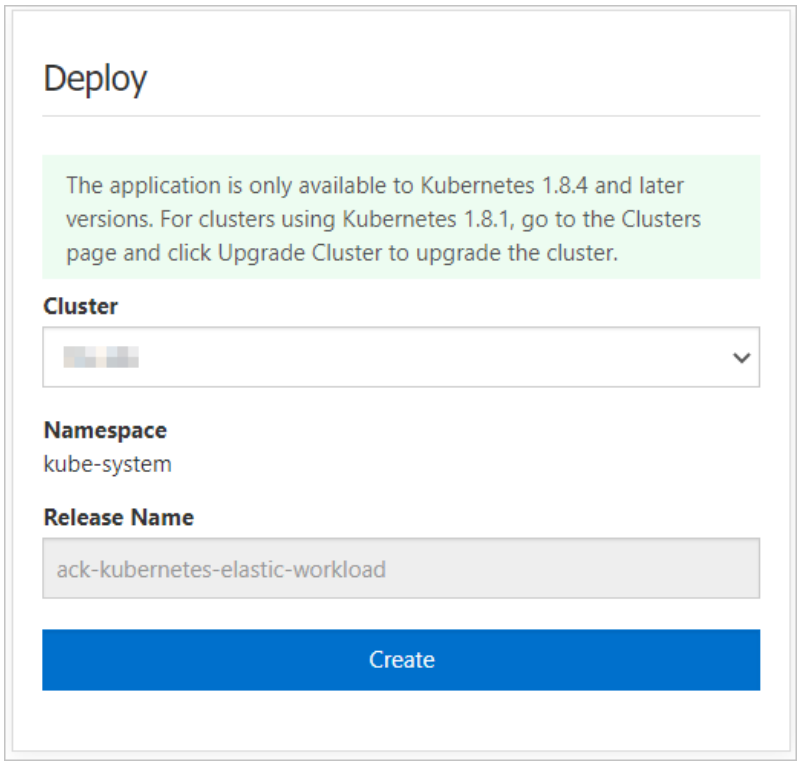
Prerequisites

- A managed Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群.
- A virtual node is deployed in the cluster. For more information, see Deploy the virtual node controller and use it to create Elastic Container Instance-based pods.

Install virtual-kubelet-autoscaler

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab, and find and click **ack-virtual-kubelet-autoscaler**.

   You can search for ack-virtual-kubelet-autoscaler on the **Alibaba Cloud Apps** tab. Enter **ack-virtual-kubelet-autoscaler** into the Name search box and click the search icon. You can also enter a keyword to perform a fuzzy match.
4. On the **App Catalog - ack-virtual-kubelet-autoscaler** page, select the created cluster in the **Deploy** section and click **Create**.



Result

1. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
2. In the left-side navigation pane of the cluster details page, choose **Applications > Helm**. You can find the ack-virtual-kubelet-autoscaler application.

22.7. Install the elastic workload component

You can use ack-kubernetes-elastic-workload to create elastic workloads. An elastic workload listens on a source workload and can create elastic units based on the scheduling policies of the elastic units. If the number of pod replicas in an elastic workload exceeds a threshold, the numbers of pod replicas scheduled to the source workload and elastic units are adjusted. This topic describes how to deploy and use ack-kubernetes-elastic-workload in a Container Service for Kubernetes (ACK) cluster.

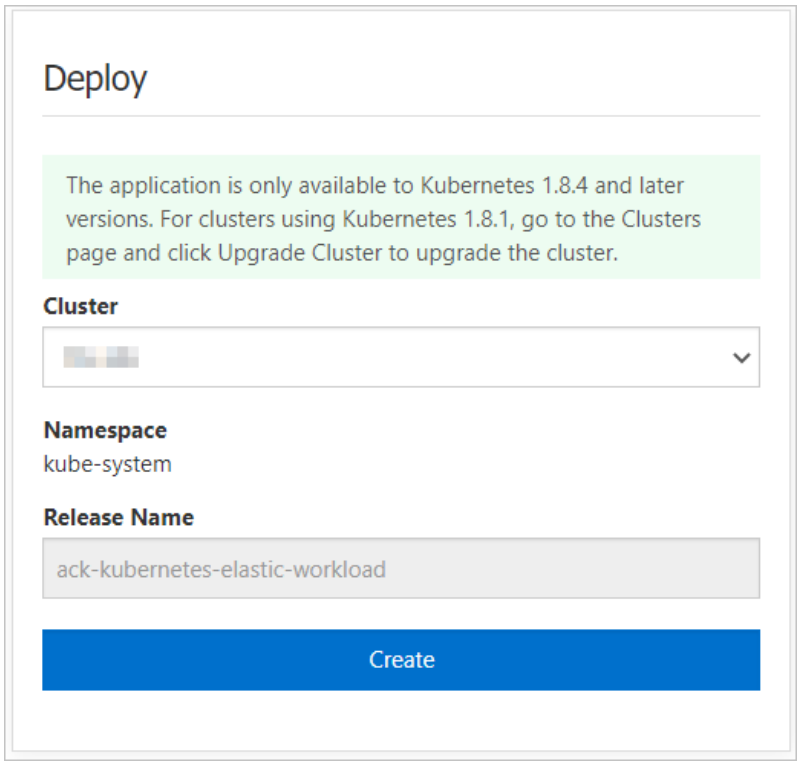
Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- ack-virtual-node is deployed in the ACK cluster. For more information, see Use Elastic Container Instance in ACK clusters.

Install ack-kubernetes-elastic-workload

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab. On the Alibaba Cloud Apps tab, find and click **ack-kubernetes-elastic-workload**.

   You can search for **ack-kubernetes-elastic-workload** on the **Alibaba Cloud Apps** tab. Enter ack-virtual-kubelet-autoscaler into the Name search box and click the search icon.
4. On the **App Catalog - ack-kubernetes-elastic-workload** tab, select the created cluster in the **Deploy** section and click **Create**.



5. In the left-side navigation pane of the ACK console, click **Clusters**.
6. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
7. In the left-side navigation pane of the details page, choose **Applications > Helm**. You can verify that ack-kubernetes-elastic-workload is deployed.

Use elastic workloads

In a Kubernetes cluster, you must deal with pod scheduling and pod lifecycle management for workloads. Pod scheduling and pod lifecycle management are based on the following operations:

- Change the pod scheduling policy when the number of pod replicas reaches a threshold.
- Prioritize specific pods for lifecycle management.

This section describes how to use elastic workloads to perform the preceding operations.

Create an application by using a Deployment based on the following template:

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
 name: nginx-deployment-basic
 labels:
  app: nginx
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
  #  nodeSelector:
  #   env: test-team
   containers:
   - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
```

Create an elastic workload for the application.

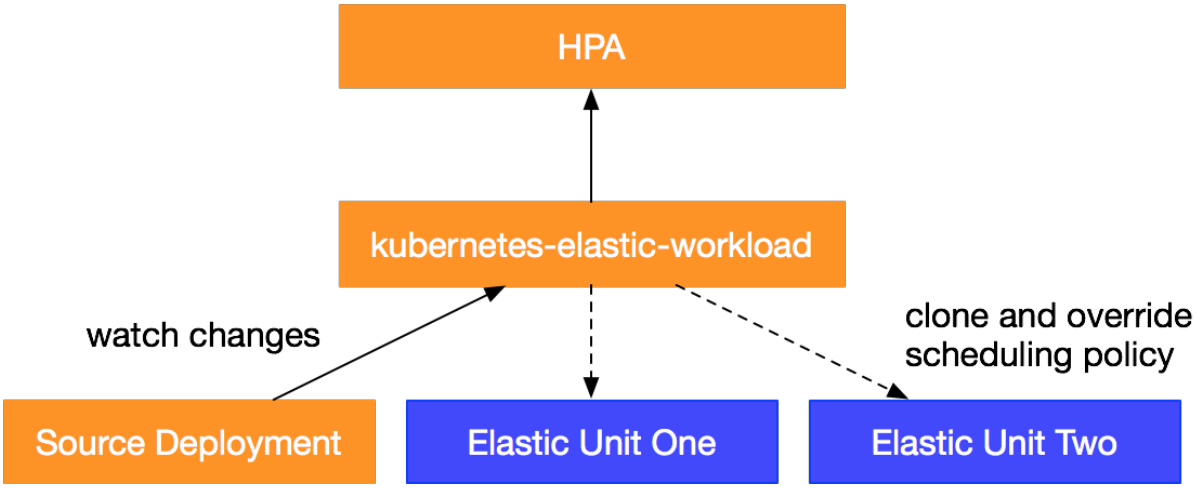
```
# Define an elastic workload.
apiVersion: autoscaling.alibabacloud.com/v1beta1
kind: ElasticWorkload
metadata:
 name: elasticworkload-sample
spec:
 sourceTarget:
  name: nginx-deployment-basic
  kind: Deployment
  apiVersion: apps/v1
  min: 2
  max: 4
 replicas: 6
 elasticUnit:
 - name: virtual-kubelet
  labels:
   virtual-kubelet: "true"
  annotations:
   virtual-kubelet: "true"
  nodeSelector:
   type: "virtual-kubelet"
  tolerations:
  - key: "virtual-kubelet.io/provider"
   operator: "Exists"
  # min: 0 You can specify different values for different elastic units.
  # max: 10
```

The preceding template shows how to create an elastic workload for the application. Similar to the use of Horizontal Pod Autoscaler (HPA), ack-kubernetes-elastic-workload is deployed as an external plug-in and does not interfere with your workloads.

Typically, the configurations of an elastic workload include the following parts:

1. SourceTarget: defines the type of the source workload and the range of the number of pod replicas.
2. elasticUnit: an array of fields that define the scheduling policies of an elastic unit. If you want to define multiple elastic units, set the fields in the same order as shown in the template for each elastic unit.

In this example, the number of pod replicas provisioned for the source workload ranges from 2 to 4. This indicates that pod replicas are scheduled to the source workload if the elastic workload has 2 to 4 pod replicas. If the number of pod replicas is greater than 4, extra pod replicas are scheduled to the virtual-kubelet elastic unit. Pod replicas are scheduled based on the scheduling policies of virtual-kubelet, such as the label, annotation, nodeSelector, affinity, and toleration parameters.



The elastic workload listens on the source workload and can create elastic units based on the scheduling policy of the elastic unit. If the number of pod replicas in the elastic workload exceeds the threshold, the numbers of pod replicas scheduled to the source workload and the elastic unit are adjusted.

After you deploy the elastic workload with the preceding template, you can run the following command to query the state of the elastic workload. The value of the Desired Replicas field in the Status section of the output indicates the number of pod replicas that are scheduled to the elastic unit.

Run the following command to query the state of the elastic workload:

```
kubectl describe ew elasticworkload-sample  # same as kubectl get elasticworkload
```

Expected output:

```
Name:       elasticworkload-sample
Namespace:  default
Labels:     &amp;amp;lt;none&amp;amp;gt;
Annotations: &amp;amp;lt;none&amp;amp;gt;
API Version: autoscaling.alibabacloud.com/v1beta1
Kind:       ElasticWorkload
Metadata:
  Creation Timestamp: 2020-05-06T03:43:41Z
  Generation:       27
  Resource Version:  20635284
  Self Link:        /apis/autoscaling.alibabacloud.com/v1beta1/namespaces/default/elasticworkloads/elasticworklo
ad-sample
  UID:              0e9205ff-38b8-43b7-9076-ffa130f26ef4
Spec:
  Elastic Unit:
    Annotations:
      Virtual - Kubelet: true
    Labels:
      Virtual - Kubelet: true
    Name:              demo
    Node Selector:
      Type: virtual-kubelet
    Tolerations:
      Key:     virtual-kubelet.io/provider
      Operator: Exists
  Replicas:    6
  Source Target:
    API Version: apps/v1
    Kind:       Deployment
    Max:       2
    Min:       0
    Name:      nginx-deployment-basic
Status:
  Elastic Units Status:
    Desired Replicas: 4
    Name:          nginx-deployment-basic-unit-virtual-kubelet
    Update Timestamp: 2020-05-07T12:38:27Z
  Replicas:        6
  Selector:        app=nginx
  Source Target:
    API Version:     apps/v1
    Desired Replicas: 2
    Kind:          Deployment
    Name:          nginx-deployment-basic
    Update Timestamp: 2020-05-07T12:38:27Z
Events:            &amp;amp;lt;none&amp;amp;gt;
```

After you deploy the elastic workload with the preceding template, you can run the following command to query the states of the pods: The output shows that new Deployments and pods are created by cloning the source workload. You can also find that the number of pod replicas in these Deployments complies with the scheduling policy.

Run the following command to query the states of the pods:

```
kubectl get pod -o wide
```

Expected output:

```
NAME                                          READY  STATUS   RESTARTS  AGE   IP       NODE           NOMINATED NODE  READINE
SS GATES
nginx-deployment-basic-7ff9955f89-djxwv       1/1    Running  0         138m  172.20.*.***  cn-hangzhou.10.0.*.***
&amp;amp;lt;none&amp;amp;gt;       &amp;amp;lt;none&amp;amp;gt;
nginx-deployment-basic-7ff9955f89-hrw2z       1/1    Running  0         138m  172.2*.*.**   cn-hangzhou.10.0.*.*** &
amp;amp;lt;none&amp;amp;gt;       &amp;amp;lt;none&amp;amp;gt;
nginx-deployment-basic-unit-demo-8bb586568-4f8xt  1/1    Running  0         138m  10.1.**.**    virtual-node-eci-1
&amp;amp;lt;none&amp;amp;gt;       &amp;amp;lt;none&amp;amp;gt;
nginx-deployment-basic-unit-demo-8bb586568-bl5pd  1/1    Running  0         138m  10.1.**.**    virtual-node-eci-0
&amp;amp;lt;none&amp;amp;gt;       &amp;amp;lt;none&amp;amp;gt;
nginx-deployment-basic-unit-demo-8bb586568-ndbp8  1/1    Running  0         138m  10.1.**.**    virtual-node-eci-0
&amp;amp;lt;none&amp;amp;gt;       &amp;amp;lt;none&amp;amp;gt;
nginx-deployment-basic-unit-demo-8bb586568-vx9jx  1/1    Running  0         138m  10.1.**.**    virtual-node-eci-2
&amp;amp;lt;none&amp;amp;gt;       &amp;amp;lt;none&amp;amp;gt;
```

You can also use HPA to scale an elastic workload, as shown in the preceding figure. If an elastic workload is scaled by HPA, the elastic workload adjusts the number of pod replicas scheduled to each elastic unit. For example, if HPA reduces the number of pod replicas in the elastic workload from 6 to 4, elastic units will scale in and the pod replicas of elastic units are reduced in priority.

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
 name: elastic-workload-demo
 namespace: default
spec:
 scaleTargetRef:
  apiVersion: autoscaling.alibabacloud.com/v1beta1
  kind: ElasticWorkload
  name: elasticworkload-sample
 minReplicas: 2
 maxReplicas: 10
 metrics:
 - type: Resource
  resource:
   name: cpu
   target:
    type: Utilization
    averageUtilization: 50
```

Note Elastic workloads create Deployments by cloning source Deployments and overwriting scheduling policies. This allows you to manage scheduling policies. Elastic workloads also adjust the pod replicas scheduled to source workloads and elastic units. This allows you to prioritize specific pods.

23.Windows container

23.1. Create a Windows node pool

Container Service for Kubernetes (ACK) allows you to use a node pool to manage multiple nodes in a cluster as a group. For example, you can centrally manage the labels and taints of the nodes in a node pool. This topic describes how to create a node pool that consists of Windows nodes in the ACK console.

Prerequisites

- An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.
- The Kubernetes version of the cluster is later than V1.9.

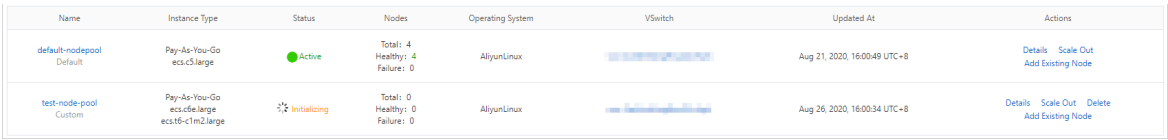
> **Notice**
>
> - A Windows node pool supports only Flannel as the network plug-in and does not support Terway.
> - By default, a cluster can contain at most 100 nodes. To increase the quota,Submit a ticket.
> - When you add an Elastic Compute Service (ECS) instance to a node pool, make sure that the ECS instance is associated with an elastic IP address (EIP) or a NAT gateway is configured for the virtual private cloud (VPC) where the ECS instance is deployed. In addition, make sure that the ECS instance can access the Internet. Otherwise, you cannot add the ECS instance.
> - Windows node pools support Windows Server 2019 and Windows Server 1909 Core.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. In the upper-right corner of the **Node Pools** page, click **Create Node Pool**.

   In the upper-right corner of the **Node Pools** page, you can also click **Create Managed Node Pool** to create a managed node pool, or click **Configure Auto Scaling** to create an auto-scaling node pool.
6. In the **Create Node Pool** dialog box, configure the node pool.

   For more information about the parameters, see 创建Kubernetes托管版集群. The following list describes some of the parameters.
   - Quantity: Specify the initial number of nodes in the node pool. If you do not want to add nodes to the pool, set this parameter to 0.
   - Operating System: Select a Windows operating system.
   - Node Label: Add labels to the nodes in the node pool.
   - ECS Label: Add labels to the ECS instances.
7. Click **Confirm Order**.
   On the **Node Pools** page, if the **state** of the node pool is **Initializing**, it indicates that the node pool is being created. After the node pool is created, the **state** of the node pool changes to **Active**.



23.2. Create a Windows application

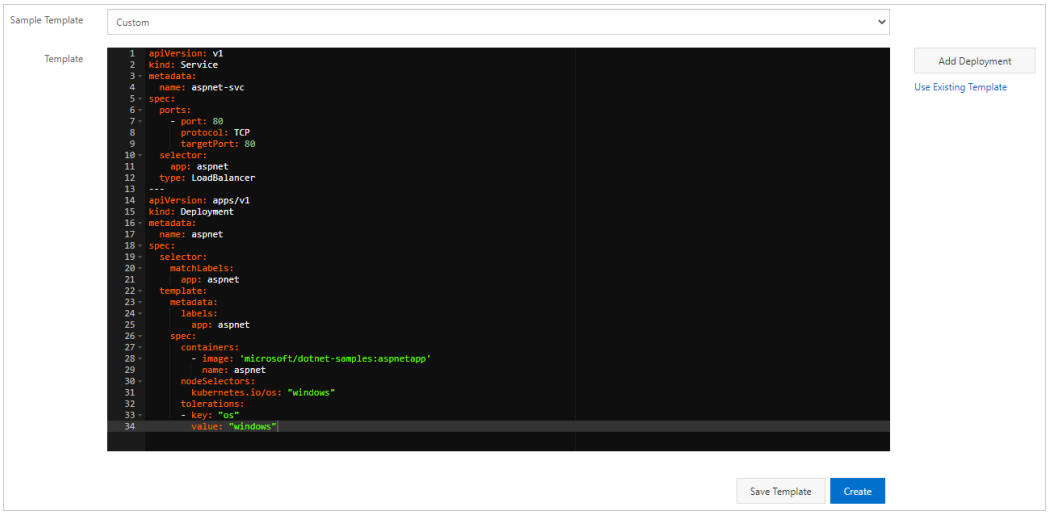
This topic describes how to create a Windows application by using an orchestration template. In this topic, a Deployment and a Service are created to run an ASP.NET application. The Deployment provisions pods for the application and the Service allows access to pods at the backend.

Prerequisites

- A Windows node is created. For more information, see Create a Windows node pool.
- The resource objects required for running an ASP.NET application are created by using an orchestration template in the Container Service for Kubernetes (ACK) console. You can provision these resource objects by using mechanisms, such as label selectors.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from YAML** in the upper-right corner.
6. Set the parameters and click **Create**.
   - **Namespace**: Select the namespace to which the resource objects belong. By default, the default namespace is selected. Most resources are scoped to namespaces, except for underlying computing resources, such as nodes and persistent volumes (PVs).
   - **Sample Template**: ACK provides YAML templates of various resource types. This simplifies the deployment of resource objects. You can also create a custom template based on YAML syntax to define the resources that you want to create.
   - **Add Deployment**: This feature allows you to define a YAML template.
   - **Use Existing Template**: You can import an existing template to the configuration page.
   - **Save As**: You can save the template that you have configured.



The following sample template is created from an orchestration template provided by ACK. This sample template provides an example on how to create a Deployment for an ASP.NET application.

> **Note** ACK supports YAML syntax. To declare multiple resource objects in a template, you can use the `---` symbol to separate the resource objects.

```
apiVersion: v1
kind: Service
metadata:
 name: aspnet-svc
spec:
 ports:
  - port: 80
    protocol: TCP
    targetPort: 80
 selector:
  app: aspnet
 type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: aspnet
spec:
 selector:
  matchLabels:
   app: aspnet
 template:
  metadata:
   labels:
    app: aspnet
  spec:
   containers:
    - image: 'mcr.microsoft.com/dotnet/samples:aspnetapp'
      name: aspnet
   nodeSelectors:
    kubernetes.io/os: "windows"
   tolerations:
   - key: "os"
    value: "windows"
```

> **Note** The image `mcr.microsoft.com/dotnet/samples:aspnetapp` is a sample image provided by Microsoft. For more information, see .NET Samples.

7. Click **Create**. A message that indicates the deployment status appears. After the deployment is complete, you can view the Deployment on the **Deployments** tab.
8. Access the ASP.NET application.
   i. In the left-side navigation pane of the ACK console, click **Clusters**.
   ii. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
   iii.

iv. On the **Services** page, you can find that a Service named **aspnet-svc** is created for the application. An external endpoint is also displayed on the page.

v. Click the external endpoint of the **aspnet-svc** Service to access the ASP.NET application.

23.3. Use Logtail to collect application logs from Windows nodes

You can use Logtail to collect application logs from Windows nodes. This topic describes how to use Logtail to collect application logs from Windows nodes.

Prerequisites

- A Container Service for Kubernetes (ACK) cluster is created and **Enable Log Service** is selected when you create the cluster. For more information, see 创建Kubernetes托管版集群.
- A Windows node pool is created. For more information, see Create a Windows node pool.

Deploy Logtail on a Windows node

1. Connect to Kubernetes clusters by using kubectl.
2. Run the following command to deploy Logtail as a DaemonSet on a Windows node:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
 labels:
  k8s-app: logtail-ds-windows
 name: logtail-ds-windows
 namespace: kube-system
spec:
 revisionHistoryLimit: 10
 selector:
  matchLabels:
   app: logtail-ds-windows
 template:
  metadata:
   annotations:
    scheduler.alpha.kubernetes.io/critical-pod: ""
   labels:
    app: logtail-ds-windows
  spec:
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: type
          operator: NotIn
          values:
          - virtual-kubelet
        - key: beta.kubernetes.io/os
          operator: In
          values:
          - windows
      - matchExpressions:
        - key: type
```

```
        operator: NotIn
        values:
        - virtual-kubelet
      - key: kubernetes.io/os
        operator: In
        values:
        - windows
  containers:
  - name: logtail
    command:
    - powershell.exe
    - -NoLogo
    - -NonInteractive
    - -File
    - entrypoint.ps1
    env:
    # --- configure the logtail file configuration generation --- #
    #
    ## "ALICLOUD_LOGTAIL_CONFIG_PATH" specifies the configuration path of logtail,
    ## if the path is blank, the configuration will generate
    ## via "ALICLOUD_LOGTAIL_CONFIG_ITEM__" prefix environment variables.
    #
    ## "ALICLOUD_LOGTAIL_CONFIG_ITEM__" needs to be combined with a type indicator to
    ## display the type of the configuration item, the following indicators are optional.
    ## -     INT__ : 64-bit signed integer.
    ## -    UINT__ : 64-bit unsigned integer.
    ## -    DOUB__ : 64-bit floating-point number.
    ## -    BOOL__ : boolean.
    ## -     STR__ : string, default type.
    ## - {<TYPE>}S__ : array in <TYPE> with vertical bar separated format.
    #
    ## P.S: don't treate "ALICLOUD_LOGTAIL_CONFIG_ITEM__" as a silver bullet,
    ## when the configuration is too complicated,
    ## please mount a detailed configuration file on "ALICLOUD_LOGTAIL_CONFIG_PATH".
    #
    - name: ALICLOUD_LOGTAIL_CONFIG_PATH
      valueFrom:
        configMapKeyRef:
          key: log-config-path
          name: alibaba-log-configuration
    - name: ALICLOUD_LOGTAIL_CONFIG_ITEM__DOUB__CPU_USAGE_LIMIT
      valueFrom:
        configMapKeyRef:
          key: cpu-core-limit
          name: alibaba-log-configuration
    - name: ALICLOUD_LOGTAIL_CONFIG_ITEM__UINT__MEM_USAGE_LIMIT
      valueFrom:
        configMapKeyRef:
          key: mem-limit
          name: alibaba-log-configuration
    - name: ALICLOUD_LOGTAIL_CONFIG_ITEM__UINT__MAX_BYTES_PER_SEC
      valueFrom:
        configMapKeyRef:
          key: max-bytes-per-sec
          name: alibaba-log-configuration
    - name: ALICLOUD_LOGTAIL_CONFIG_ITEM__UINT__SEND_REQUESTS_CONCURRENCY
      valueFrom:
```

```
      configMapKeyRef:
        key: send-requests-concurrency
        name: alibaba-log-configuration
    # --- configure the logtail configuration --- #
    #
    ## "ALICLOUD_LOG_REGION" specifies the region of the Alibaba Cloud,
    ## it will discover the region automatically if blank.
    #
    ## "ALICLOUD_LOG_USER_ID" is the same as "ALIYUN_LOGTAIL_USER_ID",
    ## which is to specify the uid of Alibaba Cloud SLS service.
    #
    ## "ALICLOUD_LOG_PROJECT" is the same as "ALICLOUD_LOG_DEFAULT_PROJECT",
    ## which is to specify the project of Alibaba Cloud SLS service.
    #
    ## "ALICLOUD_LOG_MACHINE_GROUP" is the same as "ALIYUN_LOGTAIL_USER_DEFINED_ID"
    ## and "ALICLOUD_LOG_DEFAULT_MACHINE_GROUP",
    ## which is to specify the machine group of Alibaba Cloud SLS service.
    #
    ## "ALICLOUD_LOG_ENDPOINT" specifies the endpoint of the Alibaba Cloud SLS service.
    #
    ## "ALICLOUD_LOG_ECS_FLAG" specifies whether to log the ECS flags.
    #
    ## "ALICLOUD_LOG_DOCKER_ENV_CONFIG"
    #
    ## "ALICLOUD_LOG_ENV_TAGS" is the same as "ALIYUN_LOG_ENV_TAGS",
    ## which is to specify the environment variables to be recorded,
    ## it is in form of vertical bar separated list.
    #
    - name: ALICLOUD_LOG_REGION
      value: ""
    - name: ALICLOUD_LOG_USER_ID
      valueFrom:
        configMapKeyRef:
          key: log-ali-uid
          name: alibaba-log-configuration
    - name: ALICLOUD_LOG_PROJECT
      valueFrom:
        configMapKeyRef:
          key: log-project
          name: alibaba-log-configuration
    - name: ALICLOUD_LOG_MACHINE_GROUP
      valueFrom:
        configMapKeyRef:
          key: log-machine-group
          name: alibaba-log-configuration
    - name: ALICLOUD_LOG_ENDPOINT
      valueFrom:
        configMapKeyRef:
          key: log-endpoint
          name: alibaba-log-configuration
    - name: ALICLOUD_LOG_ECS_FLAG
      value: "true"
    - name: ALICLOUD_LOG_DOCKER_ENV_CONFIG
      value: "true"
    - name: ALICLOUD_LOG_ENV_TAGS
      value: _node_name_|_node_ip_
    - name: _node_name_
```

```
        valueFrom:
          fieldRef:
            apiVersion: v1
            fieldPath: spec.nodeName
        - name: _node_ip_
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: status.hostIP
        # Replace <cn-hangzhou> in the following registry address with the ID of the region where the cluster is d
eployed.
        image: registry-vpc.cn-hangzhou.aliyuncs.com/acs/logtail-windows:v1.0.19
        imagePullPolicy: IfNotPresent
        resources:
          limits:
            cpu: 1
            memory: 1Gi
          requests:
            cpu: 100m
            memory: 256Mi
        securityContext:
          privileged: false
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - name: docker-pipe
          mountPath: \\.\pipe\docker_engine
        - name: docker-data
          mountPath: c:/ProgramData/docker
          readOnly: true
        - mountPath: c:/logtail_host
          name: root
          readOnly: true
      terminationGracePeriodSeconds: 30
      priorityClassName: system-node-critical
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      tolerations:
      - operator: Exists
      volumes:
      - name: docker-pipe
        hostPath:
          path: \\.\pipe\docker_engine
      - name: docker-data
        hostPath:
          path: c:/ProgramData/docker
          type: Directory
      - name: root
        hostPath:
          path: c:/
          type: Directory
    updateStrategy:
      rollingUpdate:
        maxUnavailable: 10%
      type: RollingUpdate
```

> **Notice** Logtail can collect only stdout files and deliver them to Log Service. In later versions, you can use Logtail to collect log files.

Example

After Logtail is deployed on the Windows node, use the following template to deploy a sample application to verify that Logtail works as expected:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
  app: logtail-test
 name: logtail-test
spec:
 replicas: 1
 template:
  metadata:
   labels:
    app: logtail-test
   name: logtail-test
  spec:
   containers:
   - name: nanoserver
    image: mcr.microsoft.com/windows/servercore:1809
    command: ["powershell.exe"]
    args: ["ping -t 127.0.0.1 -w 10000"]
    env:
   ######### Specify environment variables ###########
    - name: aliyun_logs_logtail-stdout
     value: stdout
    - name: aliyun_logs_logttail-tags
     value: tag1=v1
   ##################################
   nodeSelector:
    beta.kubernetes.io/os: windows
   tolerations:
   - effect: NoSchedule
    key: os
    operator: Equal
    value: windows
```

After the application is deployed, you can view the log data. For more information, see Query logs.

23.4. Mount Alibaba Cloud disks to Windows containers

You can use Alibaba Cloud disks as the storage resources for Windows containers that run in a Container Service for Kubernetes (ACK) cluster. This topic describes how to mount Alibaba Cloud disks to Windows containers.

Prerequisites

- Create a Windows node pool

- Connect to an ACK cluster by using kubectl

Procedure

1. Deploy the FlexVolume plug-in in the ACK cluster.
   i. Create a YAML file.

      Use the following template to create a YAML file and deploy the FlexVolume plug-in in the ACK cluster. Then, create a StorageClass in the Windows environment and update the alicloud-disk-controller component in the region of the disk that you want to mount. This allows you to mount the disk as a persistent volume (PV).
      FlexVolume is a volume plug-in that is supported by Kubernetes. For more information, see Storage overview.

      YAML template ❯

   ii. Run the `kubectl apply -f <YAML file name>.yaml` command to deploy the plug-in.
2. Create an application.
   i. Use the following YAML template to create a persistent volume claim (PVC) and a StatefulSet. This allows you to check whether the plug-in works as expected.

```
apiVersion: v1
kind: Service
metadata:
 name: nginx
 labels:
  app: nginx
spec:
 ports:
 - port: 80
  name: web
 clusterIP: None
 selector:
  app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
 name: disk-windows
spec:
 selector:
  matchLabels:
   app: nginx
 serviceName: "nginx"
 replicas: 1
 template:
  metadata:
   labels:
    app: nginx
  spec:
   tolerations:
   - effect: NoSchedule
    key: os
    operator: Equal
    value: windows
   containers:
   - name: nginx
    image: registry.cn-hangzhou.aliyuncs.com/acs/flexvolume:v1.16.9.7be0fa0-windows1809
    command: ["pwsh.exe"]
    args: ["-Command", "start-sleep 10000"]
    volumeMounts:
     - name: disk-essd
       mountPath: "C:\\data"
 volumeClaimTemplates:
 - metadata:
   name: disk-essd
  spec:
   accessModes: [ "ReadWriteOnce" ]
   storageClassName: alicloud-disk-essd-windows
   resources:
    requests:
     storage: 20Gi
```

Run the following command. If the plug-in is deployed, you can view the automatically created PV.

```
kubectl get pv
```

Expected output:

```
NAME              CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM          STORAGECLASS     REAS
ON  AGE
d-2zeh2yew2t48lu75****  20Gi     RWO           Delete          Bound   default/pvc-disk  alicloud-disk-ssd     2m4
6s
```

23.5. Mount SMB file systems to Windows containers

You can mount Server Message Block (SMB) file systems of Apsara File Storage NAS (NAS) to Windows containers that run in a Container Service for Kubernetes (ACK) cluster. This topic describes how to mount SMB file systems to Windows containers.

Prerequisites

- Create a Windows node pool.
- Connect to an ACK cluster by using kubectl.
- In the NAS console, create an SMB file system in the virtual private cloud (VPC) where the ACK cluster is deployed, and create a mount target for the SMB file system. For more information, see Mount an SMB file system on Windows.

Step 1: Deploy the FlexVolume plug-in

For more information, see the Deploy the FlexVolume plug-in in the ACK cluster section of the Mount Alibaba Cloud disks to Windows containers topic.

Step 2: Create a PV and a PVC

1. Use the following YAML template to create a persistent volume (PV) and a persistent volume claim (PVC).

   The following table describes the required parameters in the PV template.

Parameter	Description
driver	The driver that is used to mount the SMB file system. Set the value to alicloud/smb.exe.
server	The domain name of the mount target for the SMB file system. The mount target must be in the same VPC as the ACK cluster.
path	The path where the SMB file system is mounted. Set the value to \myshare or a subdirectory that starts with \myshare.
user	The username that is used to log on to a node. We recommend that you use workgroup\administrator.
password	The password that is used to log on to a node.

   YAML template that is used to create a PV | YAML template that is used to create a PVC

2. Run the `kubectl get pvc |grep pvc-smb` command to view the newly created PVC.
   The following output is returned:

```
pvc-smb         Bound   pv-smb          5Gi     RWX                     24h
```

Step 3: Deploy an application

```
apiVersion: v1
kind: PersistentVolume
metadata:
 labels:
  alicloud-pvname: pv-smb
 name: pv-smb
spec:
 accessModes:
 - ReadWriteMany
 capacity:
  storage: 5Gi
 flexVolume:
  driver: alicloud/smb.exe
  options:
   path: \myshare\test
   server: 25f3f4819c-eak52.cn-shenzhen.nas.aliyuncs.com
   user: workgroup\administrator
   password: ***
 persistentVolumeReclaimPolicy: Retain
```

1. Use the following YAML template to deploy an application.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-smb
 namespace: default
spec:
 selector:
  matchLabels:
   app: nginx-smb
 template:
  metadata:
   labels:
    app: nginx-smb
  spec:
   replicas: 2
   tolerations:
   - effect: NoSchedule
     key: os
     operator: Equal
     value: windows
   containers:
   - args:
     - -Command
     - start-sleep 10000
     command:
     - pwsh.exe
     image: registry.cn-hangzhou.aliyuncs.com/acs/flexvolume:v1.16.9.7be0fa0-windows1809
     imagePullPolicy: IfNotPresent
     name: nginx
     volumeMounts:
     - mountPath: /data
       name: pvc-nas
   restartPolicy: Always
   volumes:
   - name: pvc-nas
     persistentVolumeClaim:
       claimName: pvc-smb
```

2. Run the `kubectl get pod` command to view the state of the application.

   The following output is returned:

```
NAME           READY  STATUS   RESTARTS  AGE
nginx-smb-965fb4597-jz6fv  1/1   Running  0     95s
nginx-smb-965fb4597-zvbhk  1/1   Running  0     42s
```

   If the application is in the **Running** state, the application is created.

24.Multi-cloud and hybrid cloud management

24.1. Overview of the multi-cloud and hybrid cloud solution

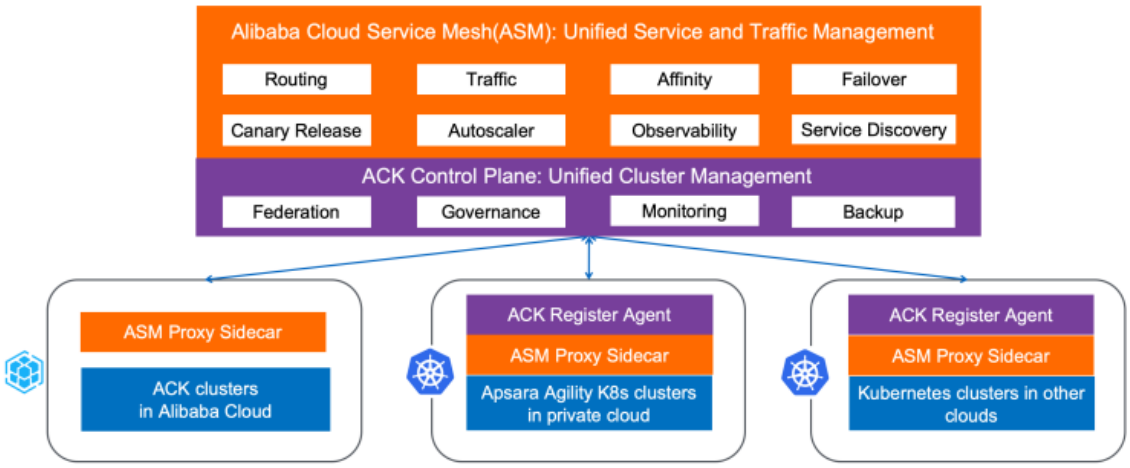
This topic describes the background and features of the multi-cloud and hybrid cloud solution provided by Container Service for Kubernetes (ACK).

Background

Cloud hosting is the trend of the future. However, some enterprise customers opt for multi-cloud or hybrid cloud solutions out of concern about data sovereignty and security. The differences in infrastructure capabilities and security architectures among different cloud environments can lead to barriers between the enterprise IT architecture and operations and maintenance (O&M) systems. This increases the complexity of multi-cloud or hybrid cloud implementations and O&M costs.

In the cloud-native era, Kubernetes-led technologies shield the differences between infrastructures and usher in the arrival of application-centric multi-cloud and hybrid cloud architectures. This facilitates application lifecycle management and resource scheduling in a unified method in multi-cloud and hybrid cloud environments.

Features



The multi-cloud and hybrid cloud solution built on ACK and Alibaba Cloud Service Mesh (ASM) provides the following features:

- Unified cluster management
  External Kubernetes clusters deployed in data centers or other clouds can be registered to the ACK console for unified management. The ACK console supports features such as security management, application management and monitoring, and logging in a unified method.
- Unified scheduling and auto scaling
  Supports auto scaling based on a unified resource scheduling framework. This allows customers to make full use of computing resources and enables auto scaling to handle unexpected traffic spikes.
- Unified service governance
  Supports nearby access, workload failover, and canary release based on ASM. This enables applications deployed among multiple clusters and across regions to support scenarios such as cloud disaster recovery and active geo-redundancy.

Security features

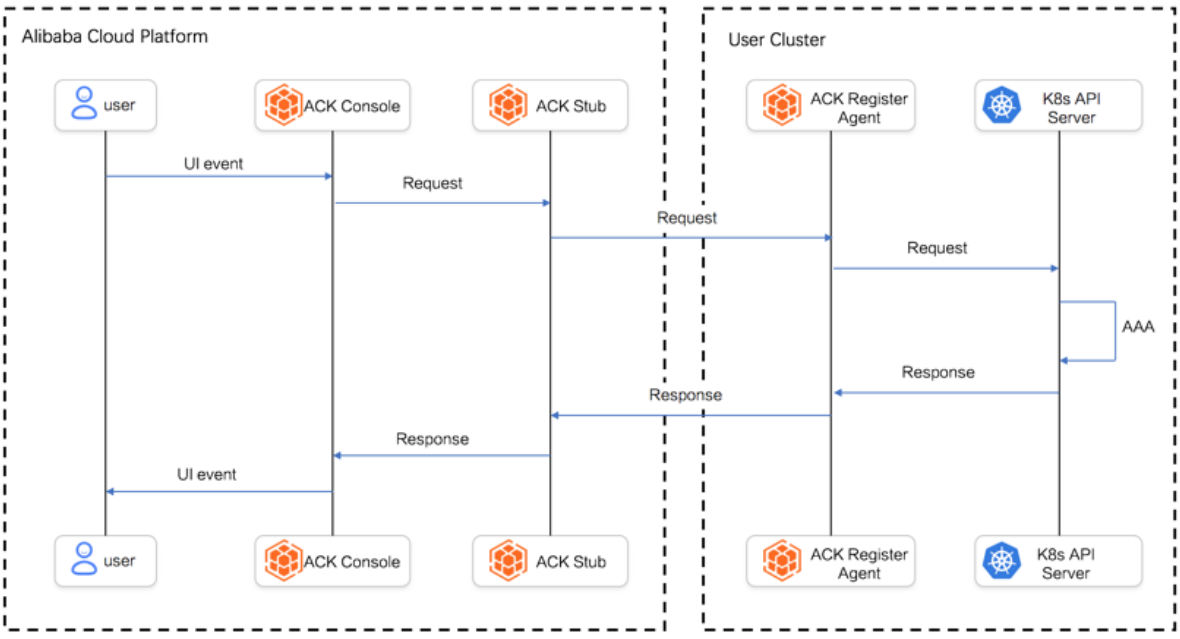
When you register external Kubernetes clusters, you can use the ACK Register Agent component to manage these clusters. This component also allows you to deploy applications in external Kubernetes clusters and manage the lifecycle of these applications.

Components

Component	Description
ACK Console	The ACK console for cluster and service management.
ACK Register Agent	ACK Register Agent is an agent that runs on Deployments within an external Kubernetes cluster. ACK Register Agent receives requests from ACK Stub and forwards them to the Kubernetes API server. ACK Register Agent also receives responses from the API server and forwards them to ACK Stub.
ACK Stub	ACK Stub is a cluster registration proxy deployed on Alibaba Cloud. ACK launches ACK Stub for all external Kubernetes clusters registered in the ACK console. ACK Stub forwards requests between the ACK console and ACK Register Agent in external Kubernetes clusters.
K8s API Server	The API server runs in an external Kubernetes cluster.

Architecture of component connections

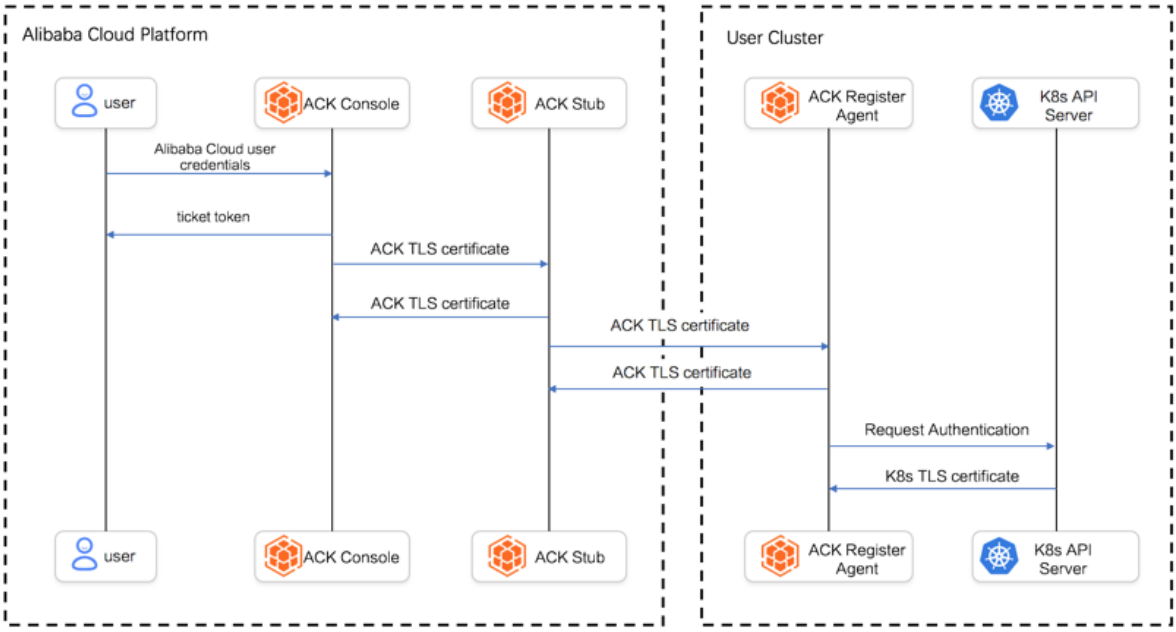
In an architecture where ACK Stub and ACK Register Agent are used, requests from the ACK console are sent to the API server in an external Kubernetes cluster. ACK Register Agent runs on Deployments that create two pods within the external Kubernetes cluster. ACK Register Agent is connected to ACK Stub that is deployed in the ACK console. The following figure shows how ACK Register Agent forwards requests to the API server within an external Kubernetes cluster.



After you register an external Kubernetes cluster in the ACK console, ACK Register Agent is deployed in the external Kubernetes cluster. Then, a two-way persistent connection is established between ACK Stub and ACK Register Agent over Transport Layer Security (TLS) 1.2. Requests from authorized users or ACK management services are first sent to ACK Stub through the TLS connection, then forwarded to ACK Register Agent, and finally delivered to the API server. After the API server receives the requests, the API server first performs authentication, authorization check, and admission control, then audits the requests, and finally returns responses. Responses are returned through the same TLS connection. They pass through ACK Register Agent and ACK Stub, and finally reach the ACK console. All requests sent to the external Kubernetes cluster through the connection are authenticated and verified. This ensures that the external Kubernetes cluster is accessed in a secure way.

Security of intercommunication among components

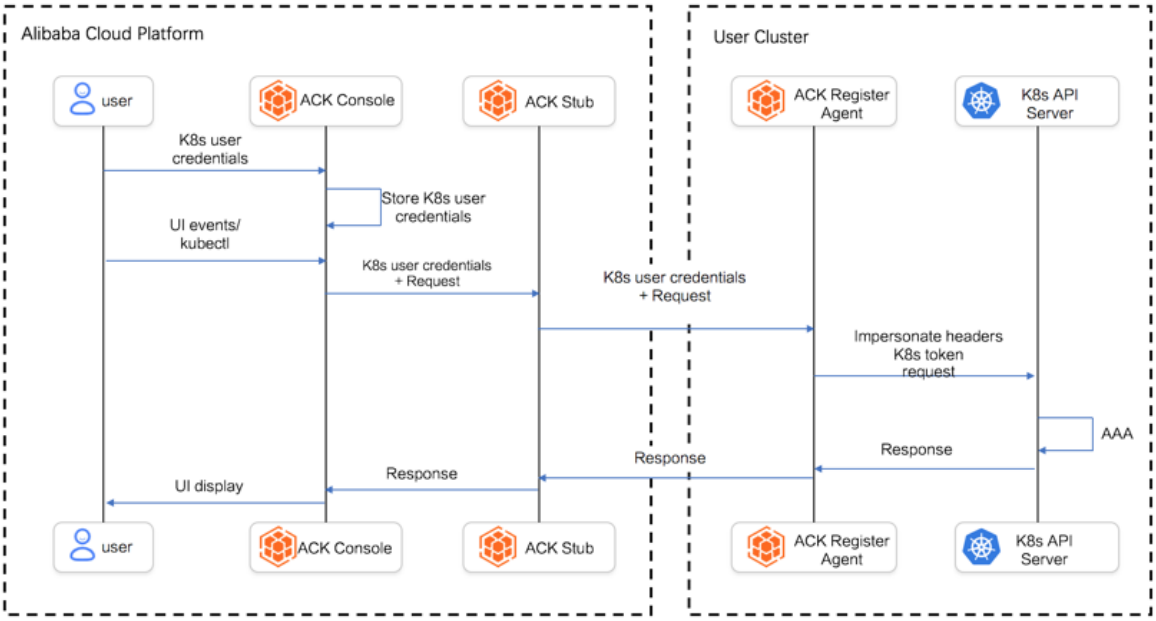
Authentication and authorization check is required for intercommunication among all components. All components to be accessed must pass security checks. This ensures that data is transmitted among only trusted components.



Authentication is based on the credentials of Resource Access Management (RAM) users and two-way TLS certification. Data is encrypted by TLS during transmission. Authorization check is based on RAM and the TLS (x509) certificate whitelisting.

- Security of request transmission
  All credentials contained in requests that are sent through the TLS connection carry the identity information of the users who send the requests. The user identity information includes the credential issued by ACK to access an external cluster and the internal credential required to access ACK components. This ensures that all requests sent to API servers are verified and audited.
- Security of user request transmission

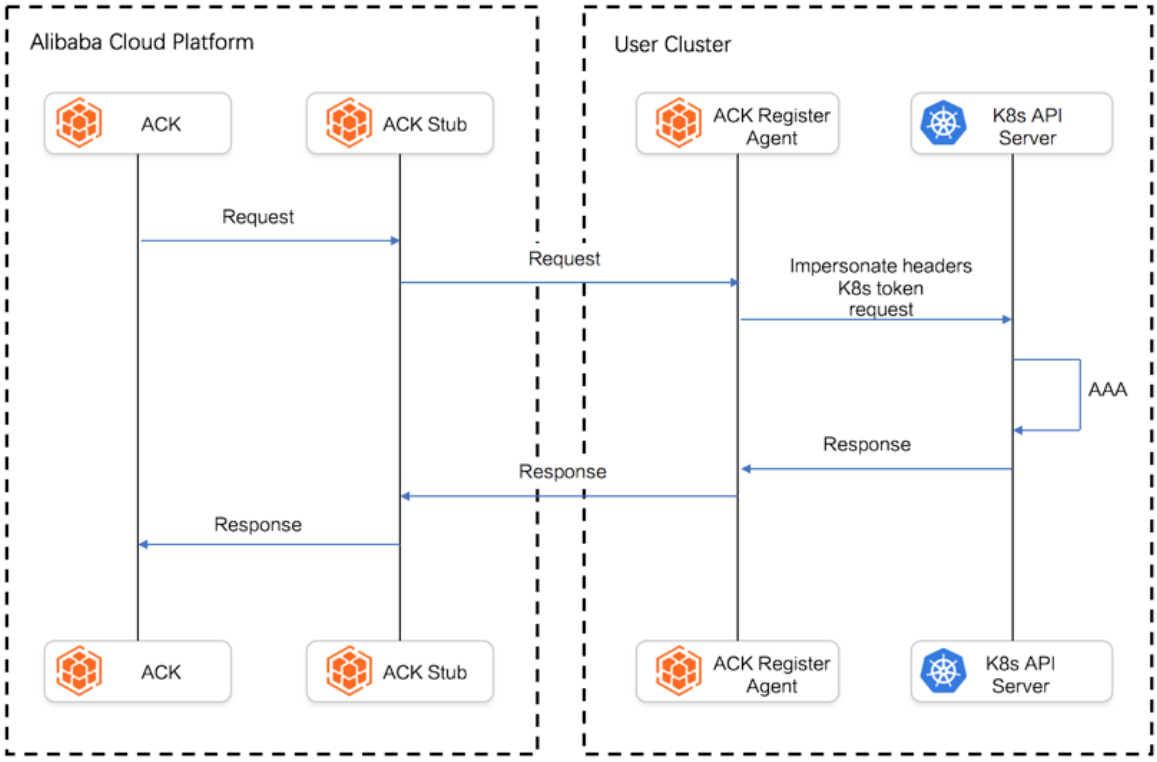
All user requests sent to external Kubernetes clusters must carry credentials issued by the ACK console. The credentials contain the user identity information. The following figure shows how a request with a credential is forwarded to an external cluster.



Cluster administrators on Alibaba Cloud can use RAM permission policies to control access from users to external clusters. Authorized RAM users can obtain credentials that are required to access external clusters from the ACK console. The credentials are provided to ACK Stub and ACK Register Agent for authentication. Data transmission among components is encrypted by TLS. After ACK Register Agent verifies the credential, the user identity carried in the credential is encapsulated into the impersonation headers of the request for the destination API server to authenticate the request. The API server performs authentication, authorization check, and admission control based on the received credential and user identity, and then audits the request.

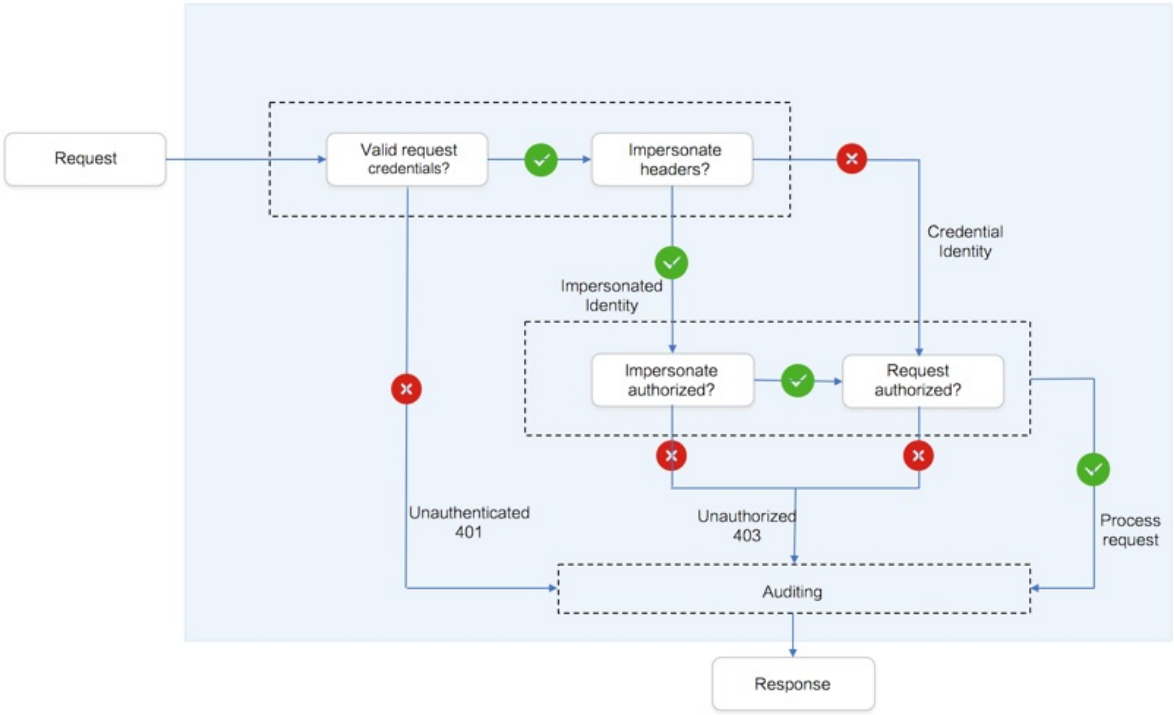
- Security of service request transmission

Requests from ACK and ASM to the external API server are transmitted in the same way as user requests. The ACK console sends a request that carries the credential required for ACK Register Agent to authenticate the request. Then, the request is authenticated by ACK Stub and ACK Register Agent and forwarded to the destination API server. During this process, ACK Register Agent impersonates the user identity in the request and forwards the request to the API server over Layer 7. Finally, the API server performs authorization checks with Role-Based Access Control (RBAC) and then audits the request. Data transmission is encrypted by TLS from end to end.



Internal security of clusters

The following figure shows how a request from ACK Register Agent is processed by an API server.



After the API server receives the request, the API server first checks whether the credential of the request is valid.

- If the credential is invalid, the API server returns a 401 error, which indicates that the request failed the authentication.
- If the request passes the authentication, the API server checks whether the request contains valid impersonation headers. If the request contains valid impersonation headers, the request is passed to the next round with the impersonated user identity.

Then, the API server checks whether the impersonated user identity is granted the required permissions on the cluster.

- If the request fails the authorization check, the API server returns a 403 error, which indicates that the user identity is unauthorized.
- If the request passes the authorization check, the API server returns a response after it audits the request.

ACK Register Agent is a forward proxy and cluster registration component written in the Go programming language. The coding and publishing of ACK Register Agent are audited by Alibaba Cloud to ensure security. The administrators of registered external clusters must ensure the security of cluster nodes by following the best practices for Kubernetes security. They can ensure the security of ACK Register Agent and external clusters by using security configurations related to infrastructure and applications.

24.2. Management of external clusters

24.2.1. Overview of registered clusters

Container Service for Kubernetes (ACK) allows you to create a cluster registration proxy and use the proxy to register a Kubernetes cluster that is deployed in a data center or on a third-party cloud. This allows you to centrally manage your clusters in the ACK console. This topic describes the features that are supported by registered clusters and the resources that are involved when you register clusters.

Benefits of registered external clusters in ACK

Your daily operations and maintenance process may involve ACK clusters, self-managed Kubernetes clusters deployed in data centers, and Kubernetes clusters deployed on third-party clouds. If multiple Kubernetes clusters are deployed in a multi-cloud or hybrid cloud architecture and you have the following requirements, we recommend that you register external clusters in ACK:

- Use the ACK console to centrally manage ACK clusters and self-managed Kubernetes clusters that are deployed in data centers. The ACK console provides the same O&M experience and security governance ability across the clusters. For example, the ACK console provides a unified logging, monitoring, and alerting system, and supports unified authorization management of Alibaba Cloud accounts and RAM users. This allows you to centrally manage all your cluster and applications.
- Use cloud computing resources to scale out self-managed Kubernetes clusters that are deployed in data centers. For example, manually or automatically adjust the number of Elastic Compute Service (ECS) instances or ECS Bare Metal instances in self-managed Kubernetes clusters that are deployed in data centers.
- Use the Application Center feature of ACK to centrally manage application lifecycles and GitOps delivery pipelines for all cloud and on-premises Kubernetes clusters. To achieve this goal, you must register on-premises Kubernetes clusters in the ACK console so that you can manage all Kubernetes clusters by using the same control plane.
- Use Alibaba Cloud Service Mesh (ASM) to implement centralized service governance for all cloud and on-premises Kubernetes clusters. To achieve this goal, you must register on-premises Kubernetes clusters in the ACK console so that you can manage all Kubernetes clusters by using the same control plane.

Comparison of features supported by registered clusters and ACK clusters

Category	Feature	ACK cluster	Registered cluster
Cluster access	Access clusters by using kubectl and kubeconfig files	✔	✔
	Access clusters by using the ACK console	✔	✔
	Manage namespaces	✔	✔
	Manage pods	✔	✔
	Manage workloads	✔	✔

Category	Feature	ACK cluster	Registered cluster
Core Kubernetes concepts	Manage persistent volumes (PVs) and storage classes	✔ Supports Alibaba Cloud disks, local disks, Apsara File Storage NAS (NAS) file systems, Cloud Paralleled File System (CPFS) file systems, and Object Storage Service (OSS) buckets.	✔ The supported storage types vary based on the environment in which the registered cluster is deployed.
	Manage Services	✔	✔
	Manage Ingresses	✔	✔
	Manage network policies	✔ Supports the Terway network plug-in.	✔ The supported network policies vary based on the environment in which the registered cluster is deployed.
	Manage ConfigMaps	✔	✔
	Manage Secrets	✔	✔
	Manage role-based access control (RBAC)	✔	✔
	Manage security policies of pods	✔	✔
	Manage resource quotas	✔	✔
	Manage Custom Resource Definitions (CRDs)	✔	✔
	Horizontal Pod Autoscaler (HPA)	✔	✔
	Pod RuntimeClasses	✔ Supports only sandboxed containers.	No.

Category	Feature	ACK cluster	Registered cluster
	Elastic container instances	✔	✔ For more information about the configurations, see Create Elastic Container Instance-based pods by using ack-virtual-node in a self-managed Kubernetes cluster.
Application management	Deploy and manage applications by using Helm charts	✔	✔
	GitOps application delivery	✔	No.
	Knative add-on	✔	✔
	Istio add-on	✔	✔
	Alibaba Cloud Service Mesh (ASM)	✔	✔ For more information about the configurations, see Use ASM to manage applications in registered external Kubernetes clusters.
Security governance	Resource Access Management (RAM)-based authentication and RBAC-based authorization	✔	✔
	Cluster auditing by using Log Service	✔	✔
	Security Center	✔	No.
	Inspections	✔	✔ For more information about the configurations, see Use the inspection feature to check for security risks in the workloads of an ACK cluster.
	Event Center (node problem detection)	✔	✔ For more information about the configurations, see Create and use a Kubernetes event center.

Category	Feature	ACK cluster	Registered cluster
Observability	Ingress charts	✔	✔ For more information about the configurations, see Monitor nginx-ingress and analyze the access log of nginx-ingress.
	Log collection	✔	✔ For more information about the configurations, see Enable Log Service for an external Kubernetes cluster.
	Application Real-Time Monitoring Service (ARMS)	✔	✔ For more information about the configurations, see Enable ARMS for an external Kubernetes cluster.
	ARMS Prometheus monitoring	✔	✔ For more information about the configurations, see Enable ARMS Prometheus for an external Kubernetes cluster.
	Architecture characteristics discovery provided by Application High Availability Service (AHAS)	✔	✔ No.
	Application throttling provided by AHAS	✔	✔ No.
	Node Problem Detector (NPD)	✔	✔ For more information about the configurations, see Create a Kubernetes event center for an external Kubernetes cluster.

Category	Feature	ACK cluster	Registered cluster
	Metrics adapter	✔	✔ For more information about the configurations, see Deploy alibaba-cloud-metrics-adapter in an external Kubernetes cluster.
	Integration with Cloud Monitor	✔	No.
	Integration with Key Management Service (KMS)	✔	No.
Cluster lifecycle management	Manage nodes	✔	✔
	Manage node pools	✔	No.
	Cluster auto scaling	✔	✔ The supported network policies vary based on the environment in which the registered cluster is deployed.
	Certificate rotation	✔	No.
	Cluster upgrades	✔	No.
	Manage system components	✔	No.
	Cluster health checks	✔	No.

Resources that are involved when you register clusters

When you register an external cluster in the ACK console, the following resources are involved:

- A virtual private cloud (VPC) and a vSwitch. The VPC must have access to the Internet. You can associate the cluster with an elastic IP address (EIP) or configure SNAT rules. You are charged for the VPC and vSwitch.
- An elastic network interface (ENI). You are not charged for the ENI.
- An internal-facing Server Load Balancer (SLB) instance. You are charged for the SLB instance based on the instance specification. For more information about the pricing of different instance specifications, see Instance types and specifications.
- (Optional)An EIP. You are charged for data transfer. For more information about data transfer fees, see Pay-as-you-go.

If you want to automatically scale self-managed on-premises Kubernetes clusters by adding or removing cloud computing resources, the following resources are also involved:

- ECS instances, ECS Bare Metal instances, and elastic container instances. You are charged for these resources based on your actual resource usage.
- Auto Scaling, which is free of charge.

You are charged other fees based on the features that you use.

24.2.2. Register an external Kubernetes cluster

This topic describes how to register and manage an external Kubernetes cluster in the Container Service for Kubernetes (ACK) console.

Prerequisites

- Resource Access Management (RAM) is activated in the RAM console. Auto Scaling (ESS) is activated in the ESS console.
- You cannot perform the following operations in the ACK console to modify a registered external Kubernetes cluster: add or remove nodes, upgrade the Kubernetes version, or modify Kubernetes components.

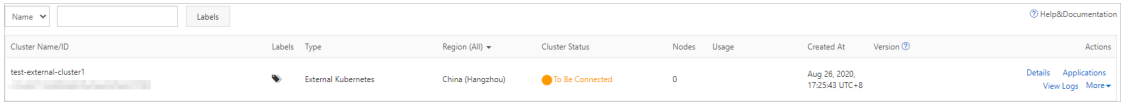
Procedure

1. Create a cluster registration proxy.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, click **Create Kubernetes Cluster** in the upper-right corner of the page.
   iv. On the **Register Cluster** tab, set the parameters.

Parameter	Description
Cluster Name	Enter a name for the ACK cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
Resource Group	Move the pointer over **All Resources** at the top of the page and select the resource group that you want to use. After you select a resource group, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource group. When you create a cluster, only the VPCs and vSwitches that belong to the selected resource group are displayed in the console. 
Region	Select a region to deploy the cluster.
Zone	Select a zone to deploy the cluster.
VPC	You can select a virtual private cloud (VPC) and a vSwitch from the drop-down lists.

Parameter	Description
Access to API Server	By default, an internal-facing Server Load Balancer (SLB) instance is created for the cluster API server. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. **Notice** If you delete the SLB instance, you cannot access the cluster API server.
EIP	Specify whether to bind an elastic IP address (EIP) to the cluster. If you select this check box, an EIP is automatically created and bound to the cluster.
Security Group	You can select **Create Basic Security Group**, **Create Advanced Security Group**, or **Select Existing Security Group**. For more information, see Overview. **Note** To select **Select Existing Security Group**, Submit a ticket to apply to be added to a whitelist.
Log Service	Specify whether to activate Log Service. You can select an existing Log Service project or create a Log Service project. If you select **Enable Log Service**, the Log Service plug-in is automatically installed in the cluster. For more information about how to set up Log Service when you deploy an application, see Collect log files from containers by using Log Service.
Deletion Protection	Specify whether to enable deletion protection. If you select this check box, the cluster cannot be deleted in the console or by calling the API. This protects the cluster from being accidentally deleted.
Labels	Add labels to the nodes in the cluster. Enter keys and values, and then click **Add**. **Note** ■ Key is required. *Value* is optional. ■ Keys are not case-sensitive. A key must not exceed 64 characters in length, and cannot start with aliyun, acs:, http://, or https://. ■ *Keys* are not case-sensitive. A key must not exceed 128 characters in length, and cannot start with aliyun, acs:, http://, or https://. ■ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the one that uses the same key. ■ You can add up to 20 labels to each resource. If you add more than 20 labels to a resource, all labels become invalid. You must remove unused labels for the remaining labels to take effect.
Terms of Service	Read and select *Terms of Service* and *Disclaimer*

v. Click **Create Cluster** on the right side of the page to deploy the cluster.
You can find the newly created cluster on the Clusters page.



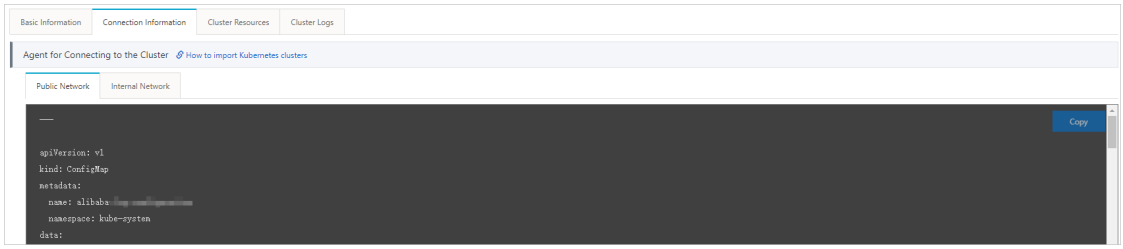
2. Register an external Kubernetes cluster.

i. On the Clusters page, find the newly created cluster and click **Details** in the **Actions** column. In this example, click cluster **test-external-cluster1**.

ii. On the **details** page of the cluster, click the **Connection Information** tab.

iii. In the **Agent for Connecting to the Cluster** section, click the **Public Network** or **Internal Network** tab based on requirements, and then click **Copy** on the right side. Create a YAML file and paste the copied code into the file. Then, use kubectl to execute the file and register the external cluster.

For example, you can create an *agent.yaml* file and paste the copied code into the *agent.yaml* file. Then, run the `kubectl apply -f agent.yaml` command on an external cluster to register the cluster.



iv. Run the `kubectl get all -n kube-system` command on the external cluster to query the agent status.

The following output is returned:

```
NAME                    READY  STATUS   RESTARTS  AGE
pod/ack-cluster-agent-655b75c987-dwp6b  1/1   Running  0     9s
NAME                DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
deployment.apps/ack-cluster-agent  1     1     1      1      26m
NAME                       DESIRED  CURRENT  READY  AGE
replicaset.apps/ack-cluster-agent-655b75c987  1     1     1    26m
```

After the external cluster is registered, go to the **Clusters** page. On the Clusters page, you can find that the cluster is in the **Running** state.

Result

On the **Clusters** page, find cluster **test-external-cluster1** and click **Details** in the **Actions** column. On the details page of the cluster, click the **Basic Information** tab to view basic information about the cluster and click the **Connection Information** tab to view information about how to connect to the cluster.

Run the `kubectl get node` command to query information about the nodes in the **test-external-cluster1** cluster. You can use kubeconfig to connect to the registered cluster and deploy applications in the cluster. For more information, see Use kubectl to connect to a Kubernetes cluster.

On the details page of the cluster, click **Releases** in the left-side navigation pane. On the **Helm** tab, you can use Helm to release and manage applications in the registered cluster.

Related information

- Overview of registered clusters

24.2.3. Create a hybrid cluster

After you register a self-managed Kubernetes cluster in the Container Service for Kubernetes (ACK) console, you can add cloud computing nodes to the cluster. This way, you can create a hybrid cluster that manages both cloud and on-premises computing resources. This topic describes how to create a hybrid cluster.

Prerequisites

- The self-managed Kubernetes cluster deployed in the data center is connected to the virtual private cloud (VPC) where the cluster registration proxy is deployed. The computing nodes and containers are connected to each other. You can use Cloud Enterprise Network (CEN) to establish the connection. For more information, see Overview.
- The self-managed Kubernetes cluster is connected to the cluster registration proxy by using the agent for internal network connection.
- The cloud computing nodes that are added through the cluster registration proxy can access the API server of the self-managed Kubernetes cluster deployed in the data center.

Procedure

1. Plan the CIDR blocks for the cluster that uses the Terway network plug-in. For more information, see Network planning.
2. Connect the on-premises Kubernetes cluster to a VPC. For more information, see Physical connection solutions.
3. Create a cluster registration proxy in the VPC and register the self-managed Kubernetes cluster. For more information, see Register an external Kubernetes cluster.
4. Configure container network plug-ins. For more information, see Install and configure container network plug-ins.
5. Create a script to add cluster nodes. For more information, see Create a script to add cluster nodes.
6. Create a node pool and add cloud computing nodes to the node pool. For more information, see Create and scale out a node pool.

   Configure auto scaling. For more information, see Configure auto scaling.

24.2.4. Install and configure container network plug-ins

The container network plug-ins used in a hybrid cluster consists of two parts: the network plug-ins that run in the data center and the network plug-ins that run on cloud computing nodes. This topic describes how to configure container network plug-ins in a hybrid cluster.

Prerequisites

In Scenario 2: The data center uses a Border Gateway Protocol (BGP) network for container networking and Scenario 3: The data center uses the host network for container networking, Terway parameters are set when you created the cluster registration proxy.

- IPVLAN is set based on business requirements.
- The number of pod vSwitches is set to an appropriate value.
- Service CIDR is set.

For more information, see Register an external Kubernetes cluster.

Scenario 1: The data center uses an overlay network for container networking

If the data center uses an overlay network for container networking, cloud computing nodes can also use this network mode. You need only to ensure that the cloud computing nodes can pull the container image used by the DaemonSet of the container network plug-in.

The following are a few commonly used overlay network modes:

- Flannel VXLAN
- Calico IPIP
- Cilium VXLAN

Scenario 2: The data center uses a Border Gateway Protocol (BGP) network for container networking

If the data center uses a Border Gateway Protocol (BGP) network for container networking, you must use the Terway network plug-in on cloud computing nodes. For more information about container networking between on-premises and the cloud, see Configure BGP.

In this scenario, make sure that the following conditions are met:

- The DaemonSet of the on-premises container network plug-in, such as BGP route reflector in Calico, must not be scheduled to cloud computing nodes.
- The DaemonSet of the Terway network plug-in must not be scheduled to on-premises computing nodes.

Each computing node that is added from a node pool through the cluster registration proxy has the `alibabacloud.com/external=true` label. You can use this label to differentiate cloud nodes from on-premises nodes.

For example, you can create `node affinity` rules to make sure that the DaemonSet of the on-premises Calico network plug-in is not scheduled to nodes that have the `alibabacloud.com/external=true` label. You can use the same method to ensure that other on-premises workloads are not scheduled to cloud computing nodes. Run the following command to update the Calico network plug-in:

```
cat <<EOF > calico-ds.pactch
spec:
 template:
  spec:
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
       - key: alibabacloud.com/external
         operator: NotIn
         values:
         - "true"
EOF
kubectl -n kube-system patch ds calico-node -p "$(cat calico-ds.pactch)"
```

By default, the DaemonSet of the Terway network plug-in is scheduled only to nodes that have the `alibabacloud.com/external=true` label.

Scenario 3: The data center uses the host network for container networking

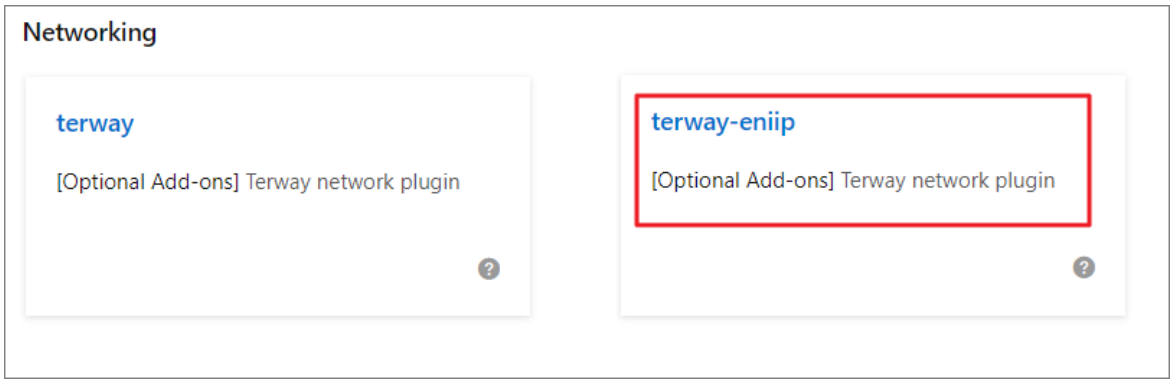
If the data center uses the host network for container networking, you need only to ensure that the DaemonSet of the Terway network plug-in is not scheduled to on-premises computing nodes. By default, the DaemonSet of the Terway network plug-in is scheduled to only nodes that have the `alibabacloud.com/external=true` label.

Install and configure the Terway network plug-in

In Scenario 1 and Scenario 2, you must install and configure the Terway network plug-in on the cloud computing nodes of the hybrid cluster.

Step 1: Install the Terway network plug-in

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. On the **Add-ons** page, click the **Networking** tab. Select the **terway-eniip** component and then click **Install**.



You can use kubeconfig to connect to the registered cluster and query the DaemonSet of the Terway network plug-in. Before cloud-computing nodes are added to the hybrid cluster, the DaemonSet will not be scheduled to on-premises nodes.
Run the following command to query the Terway network:

```
kubectl -nkube-system get ds |grep terway
```

The following output is returned:

```
terway-eniip 0     0     0     0     0     alibabacloud.com/external=true    16s
```

Step 2: Configure the Terway network plug-in

In registered clusters, all add-ons use custom AccessKey pairs to obtain the required permissions to access cloud resources.

1. Create a Resource Access Management (RAM) user and attach the following RAM permission policies to the RAM user.

   For more information, see 自定义RAM授权策略.

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "ecs:CreateNetworkInterface",
        "ecs:DescribeNetworkInterfaces",
        "ecs:AttachNetworkInterface",
        "ecs:DetachNetworkInterface",
        "ecs:DeleteNetworkInterface",
        "ecs:DescribeInstanceAttribute",
        "ecs:AssignPrivateIpAddresses",
        "ecs:UnassignPrivateIpAddresses",
        "ecs:DescribeInstances",
        "ecs:ModifyNetworkInterfaceAttribute"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "vpc:DescribeVSwitches"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

2. Run the following command to edit the *eni-config* ConfigMap and configure `eni_conf.access_key` and `eni_conf.access_secret`:

   ```
   kubectl -n kube-system edit cm eni-config
   ```

   The following *eni-config* ConfigMap is provided as an example:

```
kind: ConfigMap
apiVersion: v1
metadata:
 name: eni-config
 namespace: kube-system
data:
 eni_conf: |
 {
  "version": "1",
  "max_pool_size": 5,
  "min_pool_size": 0,
  "vswitches": {{.PodVswitchId}},
  "eni_tags": {"ack.aliyun.com":"{{.ClusterID}}"},
  "service_cidr": "{{.ServiceCIDR}}",
  "security_group": "{{.SecurityGroupId}}",
  "access_key": "",
  "access_secret": "",
  "vswitch_selection_policy": "ordered"
 }
 10-terway.conf: |
 {
  "cniVersion": "0.3.0",
  "name": "terway",
  "type": "terway"
 }
```

24.2.5. Create a script to add cluster nodes

To add nodes to a hybrid cluster, you must consider how the self-managed Kubernetes cluster is created. For example, the cluster may be created by using kubeadm, Kubernetes binaries, or Rancher. This topic describes how to create a script to add nodes to a hybrid cluster.

Step 1: Create a script to add cluster nodes

The following code of kubelet configuration is provided as an example:

```
cat >/usr/lib/systemd/system/kubelet.service <<EOF
[Unit]
Description=Kubernetes Kubelet
After=docker.service
Requires=docker.service
[Service]
ExecStart=/data0/kubernetes/bin/kubelet \\
 --node-ip=${ALIBABA_CLOUD_NODE_NAME} \\
 --hostname-override=${ALIBABA_CLOUD_NODE_NAME} \\
 --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf \\
 --config=/var/lib/kubelet/config.yaml \\
 --kubeconfig=/etc/kubernetes/kubelet.conf \\
 --cert-dir=/etc/kubernetes/pki/ \\
 --cni-bin-dir=/opt/cni/bin \\
 --cni-cache-dir=/opt/cni/cache \\
 --cni-conf-dir=/etc/cni/net.d \\
 --logtostderr=false \\
 --log-dir=/var/log/kubernetes/logs \\
 --log-file=/var/log/kubernetes/logs/kubelet.log \\
 --node-labels=${ALIBABA_CLOUD_LABELS} \\
 --root-dir=/var/lib/kubelet \\
 --provider-id=${ALIBABA_CLOUD_PROVIDE_ID} \\
 --register-with-taints=${ALIBABA_CLOUD_TAINTS} \\
 --v=4
Restart=on-failure
RestartSec=5
[Install]
WantedBy=multi-user.target
EOF
```

When you write the script, you must use the system environment variables that are provided by the external cluster registered in the ACK console. The following table lists the required system environment variables.

System environment variable	Description	Example
ALIBABA_CLOUD_PROVIDE_ID	You must set this variable in the script. Otherwise, errors may occur during cluster management.	`ALIBABA_CLOUD_PROVIDE_ID=cn-shenzhen.i-wz92ewt14n9wx9mol2cd`
ALIBABA_CLOUD_NODE_NAME	You must set this variable in the script. Otherwise, nodes in the node pool may have abnormal states.	`ALIBABA_CLOUD_NODE_NAME=cn-shenzhen.192.168.1.113`

System environment variable	Description	Example
ALIBABA_CLOUD_LABELS	You must set this variable in the script. Otherwise, errors may occur during node pool management and workload scheduling between cloud and on-premises nodes.	`ALIBABA_CLOUD_LABELS=alibabacloud.com/nodepool-id=np0e2031e952c4492bab32f512ce1422f6,ack.aliyun.com=cc3df6d939b0d4463b493b82d0d670c66,alibabacloud.com/instance-id=i-wz960ockeekr3dok06kr,alibabacloud.com/external=true,workload=cpu` The workload=cpu label is a custom label defined in the node pool configuration. Other labels are system labels.
ALIBABA_CLOUD_TAINTS	You must set this variable in the script. Otherwise, the taints that are added to the node pool do not take effect.	`ALIBABA_CLOUD_TAINTS=workload=ack:NoSchedule`

Step 2: Save the script

Save the script to an HTTP file server, such as an Object Storage Service (OSS) bucket. The sample address is `https://kubelet-liusheng.oss-ap-southeast-3-internal.aliyuncs.com/attachnode.sh`.

Step 3: Use the script

1. Register the on-premises Kubernetes cluster in the ACK console. For more information, see Register an external Kubernetes cluster.

   The cluster registration proxy automatically creates a ConfigMap named ack-agent-config in the kube-system namespace of the external cluster. The following code block shows the initial configuration of the ack-agent-config ConfigMap:

   ```
   apiVersion: v1
   data:
    addNodeScriptPath: ""
    enableNodepool: "true"
    isInit: "true"
   kind: ConfigMap
   metadata:
    name: ack-agent-config
    namespace: kube-system
   ```

2. Set the `addNodeScriptPath` field to the path of the script (`https://kubelet-liusheng.oss-ap-southeast-3-internal.aliyuncs.com/attachnode.sh`) and then save the configuration.

   The following YAML template is an example:

```
apiVersion: v1
data:
 addNodeScriptPath: https://kubelet-liusheng.oss-ap-southeast-3-internal.aliyuncs.com/attachnode.sh
 enableNodepool: "true"
 isInit: "true"
kind: ConfigMap
metadata:
 name: ack-agent-config
 namespace: kube-system
```

24.2.6. Create and scale out a node pool

You can use an elastic node pool to manage a set of Elastic Compute Service (ECS) instances and add these instances to an external Kubernetes cluster that is registered in the Container Service for Kubernetes (ACK) console. This topic describes how to create an elastic node pool for an external Kubernetes cluster.

Prerequisites

A cluster registration proxy is created and a self-managed cluster is registered in the ACK console. The self-managed cluster can be deployed in a data center or on a third-party cloud. For more information, see Register an external Kubernetes cluster.

Step 1: Create a node pool

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. In the upper-right corner of the **Node Pools** page, click **Create Node Pool**.
6. In the **Create Node Pool** dialog box, set the parameters for the node pool.

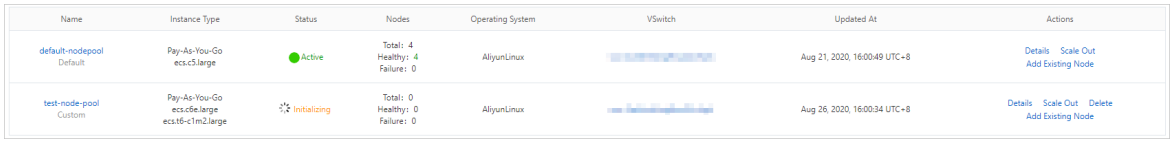
   The parameters are described in the following table.

Parameter	Description
Quantity	Specify the initial number of nodes in the node pool. If you do not want to create nodes in the node pool, set this parameter to 0.
Operating System	Select the operating system for the nodes in the node pool. Valid values: CentOS and Alibaba Cloud Linux 2.1903.
Node Label	You can add labels to the nodes in the node pool.
ECS Label	You can add labels to the ECS instances in the node pool.
Taints	You can add taints to the nodes in the node pool.
Security Group	Select the security group to which the nodes belong.

   For more information, see Create a dedicated Kubernetes cluster.

7. Click **Confirm Order**.

   On the **Node Pools** page, if the **state** of the node pool is **Initializing**, it indicates that the system is creating the node pool. After the node pool is created, the **state** of the node pool changes to **Active**.



Step 2: Scale out the node pool

Before you scale out the node pool for an external cluster, take note of the following limits:

- By default, a cluster can contain at most 100 nodes. To increase the quota, Submit a ticket.
- When you add an ECS instance to a node pool, make sure that the ECS instance is associated with an elastic IP address (EIP) or a NAT gateway is configured for the virtual private cloud (VPC) where the ECS instance is deployed. In addition, make sure that the ECS instance can access the Internet. Otherwise, you cannot add the ECS instance.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. On the Node Pools page, find the node pool that you want to scale out and click **Scale Out** in the **Actions** column.
6. In the dialog box that appears, set **Nodes to Add** to the number of nodes that you want to add and click **Submit**.

   > **Note** If you want to modify the configurations of the node pool, click **Modify Node Pool Settings**.

   On the **Node Pools** page, the **state** of the node pool changes to **Scaling**. After the scaling activity is complete, the **state** of the node pool changes to **Active**.

What to do next

1. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
2. On the Node Pools page, find the scaled node pool and click **Details** in the Actions column.
3. Click the **Nodes** tab to view detailed information about the newly added nodes.

24.2.7. Configure auto scaling

This topic describes how to enable auto scaling for a hybrid cluster.

Prerequisites

When you configure auto scaling, the cluster-autoscaler component is automatically deployed as a Deployment in the hybrid cluster. Cloud computing nodes that are added to a cluster during a scale-out event may later be removed in a scale-in event. If you install system components that run as Deployments on these nodes, the system components may not be able to provide stable services. Therefore, you must make sure that these components are not scheduled to automatically added cloud computing nodes or on-premises nodes. The following conditions must be met:

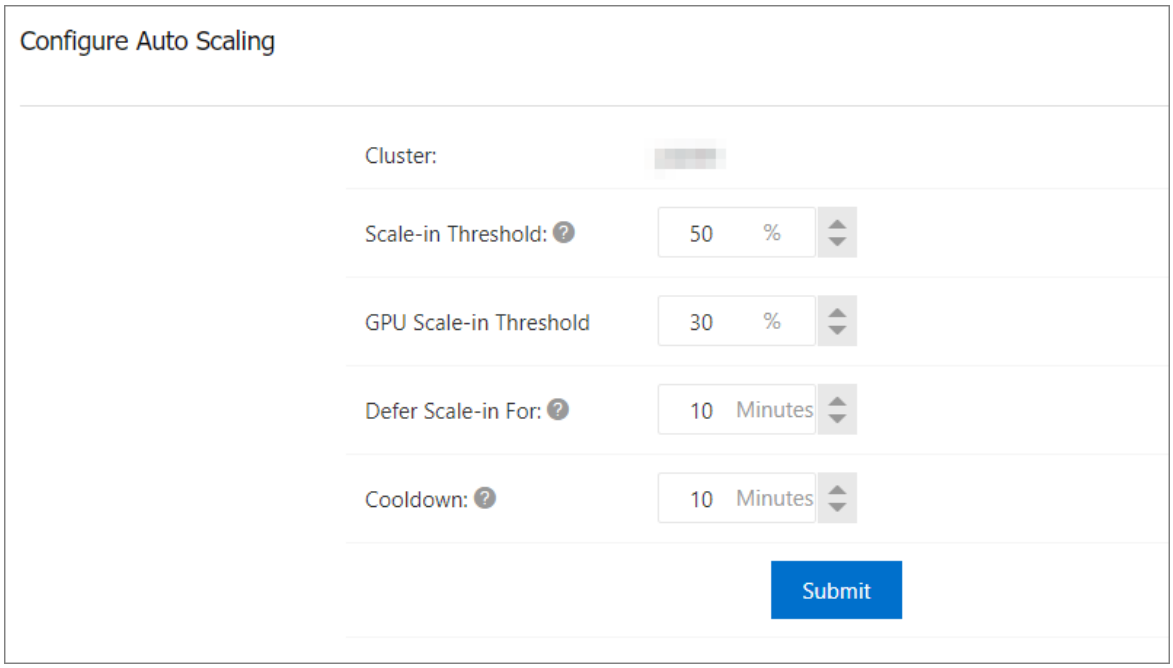
- A node pool is created and scaled out. For more information, see Create and scale out a node pool.
- The `alibabacloud.com/cloud-worker-nodes=true` label is added to nodes in the node pool. For more information, see Manage node labels.
  This ensures that the cluster-autoscaler component is automatically scheduled to a node that has the specified label.

Step 1: Configure auto scaling

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, perform the following steps to go to the **Configure Auto Scaling** page.

   You can go to the Configure Auto Scaling page in the following ways:

   - Method 1: Find the cluster that you want to manage and choose **More > Auto Scaling** in the **Actions** column.
   - Method 2:
     a. Find the cluster that you want to manage and click **Details** in the **Actions** column.
     b. In the left-side navigation pane, choose **Nodes > Node Pool**.
     c. In the upper-right corner of the **Node Pools** page, click **Configure Auto Scaling**.
4. On the **Configure Auto Scaling** page, set the parameters and then click submit.



5. Configure the cluster-autoscaler component.

   After you configure auto scaling, a Deployment of the cluster-autoscaler component is automatically created in the cluster.

   i. Run the following command to query the Deployment:

   ```
   kubectl -n kube-system get deploy |grep cluster-autoscaler
   ```

   The following output is returned:

   ```
   cluster-autoscaler      1/1   1      1      5s
   ```

ii. Configure the RAM policy that enables the cluster-autoscaler component to access other resource. The sample code is as follows:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "ess:DescribeScalingGroups",
        "ess:DescribeScalingInstances",
        "ess:DescribeScalingActivities",
        "ess:DescribeScalingConfigurations",
        "ess:DescribeScalingRules",
        "ess:DescribeScheduledTasks",
        "ess:DescribeLifecycleHooks",
        "ess:DescribeNotificationConfigurations",
        "ess:DescribeNotificationTypes",
        "ess:DescribeRegions",
        "ess:CreateScalingRule",
        "ess:ModifyScalingGroup",
        "ess:RemoveInstances",
        "ess:ExecuteScalingRule",
        "ess:ModifyScalingRule",
        "ess:DeleteScalingRule",
        "ecs:DescribeInstanceTypes",
        "ess:DetachInstances",
        "vpc:DescribeVSwitches"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

iii. Run the following commands to specify the AccessKey pair that is used to grant the RAM policy:

```
export ACCESS_KEY_ID=<ACCESS KEY ID>
export ACCESS_KEY_SECRET=<ACCESS KEY SECRET>
```

iv. Run the following command to create a Secret named *alibaba-addon-secret*:

```
kubectl -n kube-system create secret generic alibaba-addon-secret --from-literal='access-key-id=${ACCESS_KEY_ID}' --from-literal='access-key-secret=${ACCESS_KEY_SECRET}'
```

Step 2: Create a node pool for auto scaling

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, perform the following steps to go to the **Configure Auto Scaling** page.

   You can go to the Configure Auto Scaling page in the following ways:

   - Method 1: Find the cluster that you want to manage and choose **More > Auto Scaling** in the **Actions** column.
   - Method 2:

a. Find the cluster that you want to manage and click **Details** in the **Actions** column.
b. In the left-side navigation pane, choose **Nodes > Node Pool**.
c. In the upper-right corner of the **Node Pools** page, click **Configure Auto Scaling**.

4. On the **Configure Auto Scaling** page, click **Create Node Pool**.
5. In the **Create Node Pool** dialog box, set the parameters for the node pool.

   The parameters are described in the following table.

Parameter	Description
Quantity	Set the initial number of nodes in the node pool. If you do not want to create nodes in the node pool, set this parameter to 0.
Operating System	Select the operating system for the nodes. Valid values: CentOS and Alibaba Cloud Linux 2.1903.
Node Label	You can add labels to nodes in the node pool. For example, `workload=auto`.
ECS Label	You can add labels to the ECS instances in the node pool.
Taints	You can add taints to nodes in the node pool.
Security Group	Select the security group to which the nodes belong.

   For more information about the parameters, see Create a dedicated Kubernetes cluster.
6. Click **OK**.

Expected result

Run the following command to verify whether nodes that have the specified **label**, such as `workload=auto`, are added to the cluster:

```
kubectl run nginx --image nginx -l workload=auto
```

24.2.8. Pull images without a password in a self-managed Kubernetes cluster

You can use the aliyun-acr-credential-helper component to pull private images without a password from instances of Container Registry Personal Edition and Enterprise Edition in a self-managed Kubernetes cluster. This topic describes how to use aliyun-acr-credential-helper to pull a private image without a password in two scenarios.

Prerequisites

- A cluster registration proxy is created and a self-managed Kubernetes cluster is connected to the cluster registration proxy. For more information, see Register an external Kubernetes cluster.
- A kubectl client is connected to the self-managed cluster. For more information, see Connect to Kubernetes clusters by using kubectl.

Limits

aliyun-acr-credential-helper supports the following images and clusters:

- Supported images

-
- Supported clusters
  - You can use the component to pull images without a password for clusters that contain multiple namespaces.
  - The Kubernetes version of the cluster must be 1.11.2 or later. Otherwise, you must upgrade the Kubernetes version of the cluster. For more information, see Upgrade a cluster.

Precautions

- To use aliyun-acr-credential-helper, do not manually specify the imagePullSecret field. If the imagePullSecret field is specified in the template of a Kubernetes resource, such as a Deployment, the component becomes invalid.
- If a Kubernetes resource, such as a Deployment, uses custom service accounts, you must modify the service-account field in the configuration file of the component. Then, the component is authorized to pull images with the custom service accounts.
- After you create a service account in a cluster, it takes some time for aliyun-acr-credential-helper to renew the token of the service account. The new token for pulling private images is generated based on the default permissions of the ACK cluster. Applications with the service account can use the token to pull images only after the token is renewed. If you create an application immediately after you create a service account, the application will fail to pull images because it is unauthorized.
- By default, the configuration of aliyun-acr-credential-helper overwrites the imagePullSecret field of default service accounts in all namespaces. These service accounts are automatically modified when the service-account field of the **acr-configuration** ConfigMap in the kube-system namespace is changed.
- When you modify the **acr-configuration** ConfigMap in the kube-system namespace, make sure that you use the same indentation as the example in this topic. We recommend that you paste the YAML code provided in this topic to the editor, replace the corresponding values, and apply the configuration. This ensures that the format of the ConfigMap is valid.

Configure the aliyun-acr-credential-helper component in the self-managed cluster

Step 1: Configure RAM permissions for the component

Before you can install the component in an external cluster, you must set the AccessKey pair to grant the external cluster the permissions to access Alibaba Cloud resources. Before you set the AccessKey pair, create a Resource Access Management (RAM) user and grant the RAM user the permissions to access Alibaba Cloud resources.

1. Create a RAM user. For more information, see Create a RAM user.
2. Create a permission policy. For more information, see Create a custom policy.

   Use the following template to create a custom permission policy:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "cr:GetAuthorizationToken",
        "cr:ListInstanceEndpoint",
        "cr:PullRepository"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

3. Grant permissions to the RAM user. For more information, see Grant permissions to a RAM user.
4. Create an AccessKey pair for the RAM user. For more information, see Obtain an AccessKey pair.
5. Use the AccessKey pair to create a Secret named alibaba-addon-secret in the self-managed cluster.

   The system automatically uses the AccessKey pair to access cloud resources when you install the aliyun-acr-credential-helper component.

   ```
   kubectl -n kube-system create secret generic alibaba-addon-secret --from-literal='access-key-id=<your access key id>' --from-literal='access-key-secret=<your access key secret>'
   ```

   > **Note** Replace `<your access key id>` and `<your access key secret>` with the AccessKey pair that you obtained.

Step 2: Upgrade and configure the component

Before you can use the component, you may need to upgrade and configure the component.

1. Upgrade the aliyun-acr-credential-helper component.
   i. Log on to the the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and choose **More > Manage System Components** in the Actions column.
   iv. On the page that appears, find **aliyun-acr-credential-helper** in the **Security** section and click **Upgrade**.
2. Configure acr-configuration.

   Configure acr-configuration in the ACK console.

   i. Log on to the the ACK console.
   ii. In the left-side navigation pane of the ACK console, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the Actions column.
   iv. In the left-side navigation pane of the **details** page, choose **Configurations > ConfigMaps**.
   v. At the top of the **ConfigMap** page, select **kube-system** from the Namespace drop-down list. Then, find and configure **acr-configuration** by using one of the following methods:

- Method 1: Click **Edit** on the right side of acr-configuration, and enter keys and values in the **ConfigMap**.
  If **acr-configuration** is not found in the list of ConfigMaps, see Create a ConfigMap. For more information about how to update a ConfigMap, see Modify a ConfigMap.
- Method 2: Click **Edit YAML** on the right side of acr-configuration, and enter keys and values in the **ConfigMap**.

The following table describes the keys and values of the acr-configuration ConfigMap.

Key	Description	Value
service-account	The service accounts that are used by the component to pull images.	Default value: **Default**. **Note** Separate multiple service accounts with commas (,). Enter an asterisk (*) to specify all service accounts in the specified namespace.

Key	Description	Value

acr-registry-info	The information about Container Registry instances. Each instance can be specified by three string type fields in a YAML file.
	Note Set the three fields based on the following descriptions:
	■ instanceId: the ID of the Container Registry instance. This field is required for instances of Container Registry Enterprise Edition.
	■ regionId: the ID of the region where the Container Registry instance is deployed. This field is optional. The default value is the region where your ACK cluster is deployed.
	■ domains: the domain names of the Container Registry instance. This field is optional. By default, all domain names of the instance are specified. Separate multiple domain names with commas (,).
		By default, this parameter is not specified. This means that images are pulled from the default repository of the Container Registry instance that is deployed in the region where the ACK cluster is deployed.
		The following template shows the configuration of an instance of Container Registry Enterprise Edition:
		`- instanceId: cri-xxx`
		`  regionId: cn-hangzhou`
		`  domains: xxx.com,yyy.com`
		The following template shows the configuration of an instance of Container Registry Personal Edition:
		`- instanceId: ""`
		`  regionId: cn-hangzhou`
		`  domains: xxx.com,yyy.com`

watch-namespace	The namespaces to which the images to be pulled belong.	Default value: **Default**. **Note** If the value is set to all, images are pulled from all namespaces without a password. Separate multiple namespaces with commas (,).
expiring-threshold	The duration after which the cache token expires.	Default value: **15m**. We recommend that you use the default value.

Configure acr-configuration by using kubectl

i. Run the following command to open the ConfigMap of acr-configuration:

```
kubectl edit cm acr-configuration -n kube-system
```

ii. Configure the parameters of acr-configuration based on your requirements.

The following templates show the configurations of acr-configuration for instances of Container Registry Enterprise Edition and Personal Edition:

- Enterprise Edition

```
apiVersion: v1
data:
 acr-api-version: "2018-12-01"
 acr-registry-info: |-
   - instanceId: "cri-xxx"
     regionId: "cn-hangzhou"
 expiring-threshold: 15m
 service-account: default
 watch-namespace: all
kind: ConfigMap
metadata:
 name: acr-configuration
 namespace: kube-system
 selfLink: /api/v1/namespaces/kube-system/configmaps/acr-configuration
```

- Personal Edition

```
apiVersion: v1
data:
 acr-api-version: "2018-12-01"
 acr-registry-info: |-
   - instanceId: ""
     regionId: "cn-hangzhou"
 expiring-threshold: 15m
 service-account: default
 watch-namespace: all
kind: ConfigMap
metadata:
 name: acr-configuration
 namespace: kube-system
 selfLink: /api/v1/namespaces/kube-system/configmaps/acr-configuration
```

Scenario 1: Pull private images from instances of Container Registry Personal Edition and Enterprise Edition

ACK allows you to pull private images concurrently from Container Registry Enterprise Edition and Personal Edition, only from Container Registry Enterprise Edition, or only from Container Registry Personal Edition. Modify the **acr-configuration** ConfigMap based on your business requirements. For more information, see Configure the component. The following code block shows a sample configuration:

- Pull private images from Container Registry Enterprise Edition.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
   - instanceId: "cri-xxx"
     regionId: "cn-hangzhou"
     domains: "xxx.com","yyy.com"
```

- Pull private images from Container Registry Personal Edition.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
   - instanceId: ""
     regionId: "cn-hangzhou"
     domains: "xxx.com","yyy.com"
```

- Pull private images from both Container Registry Enterprise Edition and Personal Edition.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
   - instanceId: ""
   - instanceId: "cri-xxxx"
```

Scenario 2: Pull images across regions

If you want to pull images from Container Registry instances that are deployed in different regions, you must modify the **acr-configuration** ConfigMap.

For example, you want to pull images from Container Registry instances that are deployed in the China (Beijing) and China (Hangzhou) regions at the same time. In this case, modify acr-configuration as shown in the following code block. For more information, see Configure the component.

```
data:
  service-account: "default"
  watch-namespace: "all"
  expiring-threshold: "15m"
  notify-email: "cs@aliyuncs.com"
  acr-registry-info: |
   - instanceId: ""
     regionId: cn-beijing
   - instanceId: ""
     regionId: cn-hangzhou
```

24.2.9. Create Elastic Container Instance-based pods by using ack-virtual-node in a self-managed Kubernetes cluster

Virtual nodes enable seamless integration between Kubernetes and Elastic Container Instance. Virtual nodes empower Kubernetes clusters with high elasticity. This way, Kubernetes clusters are no longer limited by the computing capacity of cluster nodes. You can dynamically create Elastic Container Instance-based pods to meet your business requirements. This saves the trouble of cluster sizing. This topic describes virtual nodes and elastic container instances. This topic also describes how to create Elastic Container Instance-based pods by using ack-virtual-node.

Prerequisites

- A cluster registration proxy is created and a self-managed cluster of Kubernetes version later than 1.14 is connected to the cluster registration proxy. For more information, see Register an external Kubernetes cluster.
- Elastic Container Instance is activated. You can log on to the Elastic Container Instance console to activate the service.
- The region where the cluster is deployed must be supported by Elastic Container Instance. To view the supported regions and zones, log on to the Elastic Container Instance console.

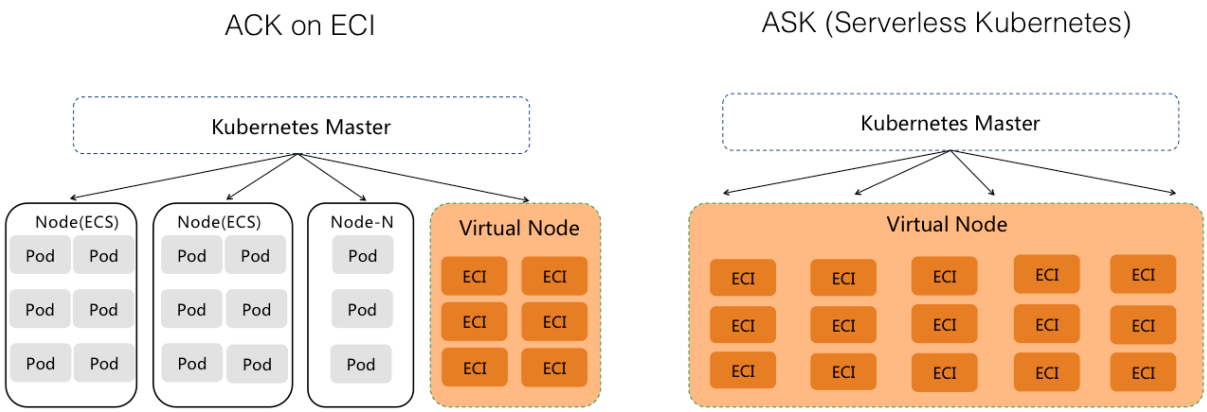
Virtual nodes and elastic container instances

Elastic Container Instance is a serverless compute service that is provided by Alibaba Cloud for containerization. You can use elastic container instances to set up an operations and maintenance (O&M)-free and isolated runtime environment for your containers. Elastic container instances allow you to focus on containerized applications without the need to purchase or manage Elastic Compute Service (ECS) instances. This way, you do not need to perform infrastructure maintenance. You can create elastic container instances to meet your business requirements. You are charged for resource usage on a per second basis.

Virtual nodes enable seamless integration between Kubernetes and Elastic Container Instance. Virtual nodes empower Kubernetes clusters with high elasticity. This way, Kubernetes clusters are no longer limited by the computing capacity of cluster nodes. You can dynamically create Elastic Container Instance-based pods to meet your business requirements. This saves the trouble of cluster sizing. Virtual nodes can significantly reduce computing costs and improve cluster elasticity in the following scenarios:

- Online business that requires elastic scaling to withstand traffic fluctuations, such as online education and e-commerce. Virtual nodes optimize the maintenance of resource pools. This can help you reduce computing costs.
- Virtual nodes can reduce costs in computing scenarios where Spark or Presto is used to process data.
- CI/CD pipeline: Jenkins and Gitlab-Runner.
- Jobs: Jobs in Artificial Intelligence (AI) computing scenarios and CronJobs.

Based on virtual nodes and elastic container instances, ACK provides multiple serverless container services, such as serverless Kubernetes (ASK) and ACK on Elastic Container Instance. You can use these services to deploy elastic and maintenance-free workloads.



Step 1: Grant a RAM user the permissions to access ack-virtual-node

Before you can install ack-virtual-node in a registered external cluster, you must specify the AccessKey information to authenticate requests that are sent to cloud resources. Before you specify the AccessKey information, create a Resource Access Management (RAM) user and grant the RAM user the permissions to access Alibaba Cloud resources.

1. Create a RAM user. For more information, see Create a RAM user.
2. (Optional)Create a custom permission policy.

   For more information about how to create a custom permission policy, see Create a custom policy. Use the following template to create a custom permission policy:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "eci:CreateContainerGroup",
        "eci:DeleteContainerGroup",
        "eci:DescribeContainerGroups",
        "eci:DescribeContainerLog",
        "eci:UpdateContainerGroup",
        "eci:UpdateContainerGroupByTemplate",
        "eci:CreateContainerGroupFromTemplate",
        "eci:RestartContainerGroup",
        "eci:ExportContainerGroupTemplate",
        "eci:DescribeContainerGroupMetric",
        "eci:DescribeMultiContainerGroupMetric",
        "eci:ExecContainerCommand",
        "eci:CreateImageCache",
        "eci:DescribeImageCaches",
        "eci:DeleteImageCache"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

3. Grant permissions to the RAM user. For more information, see Grant permissions to a RAM user.

   You can create a custom permission policy or select the AliyunECIFullAccess permission policy to authorize the RAM user.

4. Create an AccessKey pair for the RAM user. For more information, see Obtain an AccessKey pair.

5. Run the following command to create a Secret named *alibaba-addon-secret* in the registered external cluster:

   When you install ack-virtual-node in the registered external cluster, the AccessKey pair is automatically referenced to access Alibaba Cloud resources.

```
kubectl -n kube-system create secret generic alibaba-addon-secret --from-literal='access-key-id=<your access key id>' --from-literal='access-key-secret=<your access key secret>'
```

> **Note** Set `<your access key id>` and `<your access key secret>` to the AccessKey pair that you obtained.

Step 2: Install ack-virtual-node in the registered external cluster

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Operations > Add-ons**.
5. Click the **Others** tab, find ack-virtual-node, and then click **Install**.

The default vSwitch and security group of the cluster are used for elastic container instances that are deployed by ack-virtual-node. If you want to modify these settings, see Related operations.

Step 3: Create Elastic Container Instance-based pods

You can use one of the following methods to create an Elastic Container Instance-based pod in the registered external cluster:

- **Add a label to the pod.**
  Add the `alibabacloud.com/eci=true` label to the pod that you want to create. Then, an Elastic Container Instance-based pod is created and scheduled to a virtual node. Examples:
  i. Run the following command to add a label to the pod:

```
kubectl run nginx --image nginx -l alibabacloud.com/eci=true
```

  ii. Run the following command to view the pod:

```
kubectl get pod -o wide|grep virtual-kubelet
```

  Expected output:

```
nginx-7fc9f746b6-r4xgx   0/1   ContainerCreating  0     20s  192.168.XX.XX  virtual-kubelet    <none>
<none>
```

- **Add a label to the namespace of the pod.**
  Add the `alibabacloud.com/eci=true` label to the namespace to which the pod belongs. Then, an Elastic Container Instance-based pod is created and scheduled to the virtual node. Example:
  i. Run the following command to create a virtual node:

```
kubectl create ns vk
```

  ii. Run the following command to add a label to the namespace to which the pod belongs:

```
kubectl label namespace vk alibabacloud.com/eci=true
```

  iii. Run the following command to schedule the pod to the virtual node:

```
kubectl -n vk run nginx --image nginx
```

  iv. Run the following command to view the pod:

```
kubectl -n vk get pod -o wide|grep virtual-kubelet
```

  Expected output:

```
nginx-6f489b847d-vgj4d   1/1   Running      0     1m  192.168.XX.XX  virtual-kubelet    <none>      <no
ne>
```

What to do next

Modify the configurations of the virtual node

The configurations of the virtual node controller determine how Elastic Container Instance-based pods are scheduled to a virtual node and specify the runtime environment of the pod, such as vSwitches and security group settings. You can modify the configurations of the virtual node controller to meet your business requirements. Modified configurations apply to only pods that are scheduled after modifications and do not apply to existing pods that run on the node.

Run the following command to modify the configurations of the virtual node controller:

```
kubectl -n kube-system edit deployment ack-virtual-node-controller
```

The following operations are frequently performed to modify the configurations of the virtual node controller:

- **Upgrade the version of the virtual node controller**
  To use the latest features of virtual nodes, you must upgrade the virtual node controller to the latest version.
- **Modify security group settings (ECI_SECURITY_GROUP)**
  You can modify the ECI_SECURITY_GROUP environment variable to change the security group of the pods that are scheduled to the virtual node.
- **Modify vSwitch settings (ECI_VSWITCH)**
  You can modify the ECI_VSWITCH environment variable to change the vSwitch of the pods that are scheduled to the virtual node. We recommend that you configure multiple vSwitches that are deployed in different zones to ensure high availability. When elastic container instances in the current zone are out of stock, the virtual node controller creates pods in another zone.
- **Modify kube-proxy settings (ECI_KUBE_PROXY).**
  By default, the ECI_KUBE_PROXY environment variable is set to true. This indicates that pods can access ClusterIP Services. If the pods no longer need to access ClusterIP Services, you can set the environment variable to **false** to disable kube-proxy. In large-scale deployment scenarios, a cluster may need to start a large number of pods. This significantly increases the number of concurrent connections between kube-proxy and the Kubernetes API server. In this case, you can disable kube-proxy to reduce the heavy loads on the API server.
- **Modify the kube-system/eci-profile ConfigMap.**
  You can modify the kube-system/eci-profile ConfigMap to specify more parameters for elastic container instances, such as vSwitches and security groups.

Delete a virtual node

1. Uninstall ack-virtual-node.
   After you delete all the pods in a registered external cluster, you can uninstall ack-virtual-node on the **Add-ons** page.
2. Run the `kubectl delete no` command to delete related virtual nodes.

> **Note** If you do not delete the Elastic Container Instance-based pods in the cluster before you uninstall ack-virtual-node, the elastic container instances are retained in the cluster.

Related information

- Overview of registered clusters
- Deploy the virtual node controller and use it to create Elastic Container Instance-based pods
- Run a job by using a virtual node

24.3. Observability of external clusters

24.3.1. Enable Log Service for an external Kubernetes cluster

You can enable Log Service for external Kubernetes clusters that are registered in the Container Service for Kubernetes (ACK) console. This way, you can manage Kubernetes clusters that are deployed across regions in a centralized manner. This topic describes how to enable Log Service for a registered external Kubernetes cluster.

Prerequisites

An external Kubernetes cluster is registered in the ACK console. For more information, see Register an external Kubernetes cluster.

Step 1: Configure RAM permissions for the Log Service component

Before you can install the component in an external cluster, you must set the AccessKey pair to grant the external cluster the permissions to access Alibaba Cloud resources. Before you set the AccessKey pair, create a Resource Access Management (RAM) user and grant the RAM user the permissions to access Alibaba Cloud resources.

1. Create a RAM user. For more information, see Create a RAM user.
2. Create a permission policy. For more information, see Create a custom policy.

   The following code block shows the content of the permission policy for the Logtail component:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:CreateProject",
        "log:GetProject",
        "log:DeleteProject",
        "log:CreateLogStore",
        "log:GetLogStore",
        "log:UpdateLogStore",
        "log:DeleteLogStore",
        "log:CreateConfig",
        "log:UpdateConfig",
        "log:GetConfig",
        "log:DeleteConfig",
        "log:CreateMachineGroup",
        "log:UpdateMachineGroup",
        "log:GetMachineGroup",
        "log:DeleteMachineGroup",
        "log:ApplyConfigToGroup",
        "log:GetAppliedMachineGroups",
        "log:GetAppliedConfigs",
        "log:RemoveConfigFromMachineGroup",
        "log:CreateIndex",
        "log:GetIndex",
        "log:UpdateIndex",
        "log:DeleteIndex",
        "log:CreateSavedSearch",
        "log:GetSavedSearch",
        "log:UpdateSavedSearch",
        "log:DeleteSavedSearch",
        "log:CreateDashboard",
        "log:GetDashboard",
        "log:UpdateDashboard",
        "log:DeleteDashboard",
        "log:CreateJob",
        "log:GetJob",
        "log:DeleteJob",
        "log:UpdateJob",
        "log:PostLogStoreLogs",
        "log:CreateSortedSubStore",
        "log:GetSortedSubStore",
```

```
        "log:ListSortedSubStore",
        "log:UpdateSortedSubStore",
        "log:DeleteSortedSubStore",
        "log:CreateApp",
        "log:UpdateApp",
        "log:GetApp",
        "log:DeleteApp",
        "cs:DescribeTemplates",
        "cs:DescribeTemplateAttribute"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

3. Grant permissions to the RAM user. For more information, see Grant permissions to a RAM user.
4. Create an AccessKey pair for the RAM user. For more information, see Obtain an AccessKey pair.
5. Use the AccessKey pair to create a Secret named `alibaba-addon-secret` in the registered external cluster.

   Run the following command to create the Secret. The Logtail component uses the Secret.

   ```
   kubectl -n kube-system create secret generic alibaba-addon-secret --from-literal='access-key-id=<your AccessKey ID>' --from-literal='access-key-secret=<your AccessKey Secret>'
   ```

   > **Note** Replace `<your AccessKey ID>` and `<your AccessKey Secret>` with the AccessKey pair that you obtained.

Step 2: Install the logtail-ds component

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and choose **More > Manage System Components** in the **Actions** column.
4. Click the **Logs and Monitoring** tab. Find the **logtail-ds** component and click **Install**.
5. In the **Note** dialog box, click **OK**.

Step 3: Configure Log Service

For more information about how to configure Log Service when you create an application, see Step 2: Configure Log Service when you create an application.

Related information

- Overview of registered clusters
-

24.3.2. Create a Kubernetes event center for an external Kubernetes cluster

You can create a Kubernetes event center to record events of an external Kubernetes cluster. These events include changes to pod configurations and component exceptions. The Kubernetes event center collects, stores, and visualizes cluster events in real time. The event center allows you to query and analyze the events and configure alerts. This topic describes how to create a Kubernetes event center for a registered external Kubernetes cluster.

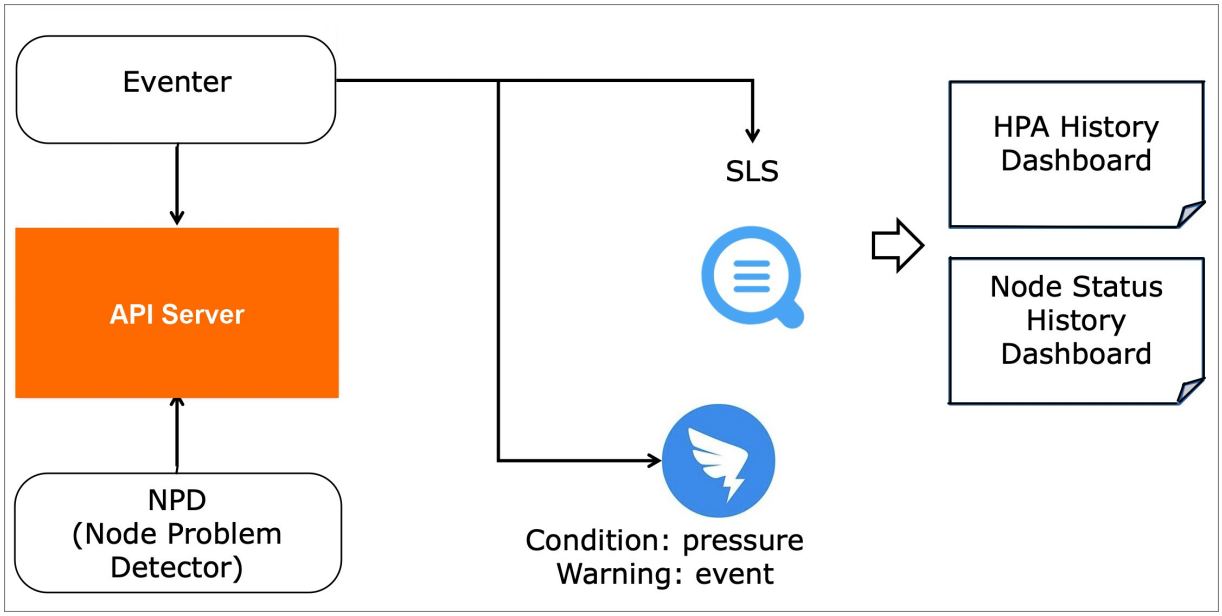
Prerequisites

An external Kubernetes cluster is registered in the ACK console. For more information, see Register an external Kubernetes cluster.

Context

Kubernetes is designed based on the state machine. Events are generated due to transitions between different states. Typically, Normal events are generated when the state machine changes to expected states and Warning events are generated when the state machine changes to unexpected states.

Container Service for Kubernetes (ACK) provides out-of-the-box monitoring solutions for events in different scenarios. The node-problem-detector and kube-eventer tools that are maintained by ACK allow you to monitor Kubernetes events.



- node-problem-detector is a tool for diagnosing Kubernetes nodes. node-problem-detector detects node exceptions, generates node events, and works with kube-eventer to trigger alerts for these events. node-problem-detector generates node events when the following exceptions are detected: Docker engine hangs, Linux kernel hangs, outbound traffic errors, and file descriptor errors. For more information, see NPD.
- kube-eventer is an open source event emitter that is maintained by ACK. kube-eventer sends Kubernetes events to sinks such as DingTalk, Log Service, and EventBridge. kube-eventer also provides filter conditions to filter different levels of events. You can use kube-eventer to collect events in real time, trigger alerts upon specific events, and asynchronously archive events. For more information, see kube-eventer.

Step 1: Configure RAM permissions for the event center component

Before you can install the component in an external cluster, you must set the AccessKey pair to grant the external cluster the permissions to access Alibaba Cloud resources. Before you set the AccessKey pair, create a Resource Access Management (RAM) user and grant the RAM user the permissions to access Alibaba Cloud resources.

1. Create a RAM user. For more information, see Create a RAM user.

2. Create a permission policy. For more information, see Create a custom policy.

   The following code block shows the content of the permission policy for the Logtail component:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:CreateProject",
        "log:GetProject",
        "log:DeleteProject",
        "log:CreateLogStore",
        "log:GetLogStore",
        "log:UpdateLogStore",
        "log:DeleteLogStore",
        "log:CreateConfig",
        "log:UpdateConfig",
        "log:GetConfig",
        "log:DeleteConfig",
        "log:CreateMachineGroup",
        "log:UpdateMachineGroup",
        "log:GetMachineGroup",
        "log:DeleteMachineGroup",
        "log:ApplyConfigToGroup",
        "log:GetAppliedMachineGroups",
        "log:GetAppliedConfigs",
        "log:RemoveConfigFromMachineGroup",
        "log:CreateIndex",
        "log:GetIndex",
        "log:UpdateIndex",
        "log:DeleteIndex",
        "log:CreateSavedSearch",
        "log:GetSavedSearch",
        "log:UpdateSavedSearch",
        "log:DeleteSavedSearch",
        "log:CreateDashboard",
        "log:GetDashboard",
        "log:UpdateDashboard",
        "log:DeleteDashboard",
        "log:CreateJob",
        "log:GetJob",
        "log:DeleteJob",
        "log:UpdateJob",
        "log:PostLogStoreLogs",
        "log:CreateSortedSubStore",
        "log:GetSortedSubStore",
        "log:ListSortedSubStore",
        "log:UpdateSortedSubStore",
        "log:DeleteSortedSubStore",
        "log:CreateApp",
        "log:UpdateApp",
        "log:GetApp",
        "log:DeleteApp",
        "cs:DescribeTemplates",
        "cs:DescribeTemplateAttribute"
      ],
      "Resource": [
```

```
        "Resource": [
          "*"
        ],
        "Effect": "Allow"
      }
    ]
}
```

3. Grant permissions to the RAM user. For more information, see Grant permissions to a RAM user.

   You can create a custom permission policy or select the AliyunECIFullAccess permission policy to authorize the RAM user.

4. Create an AccessKey pair for the RAM user. For more information, see Obtain an AccessKey pair.

5. Use the AccessKey pair to create a Secret named `alibaba-addon-secret` in the registered external cluster.

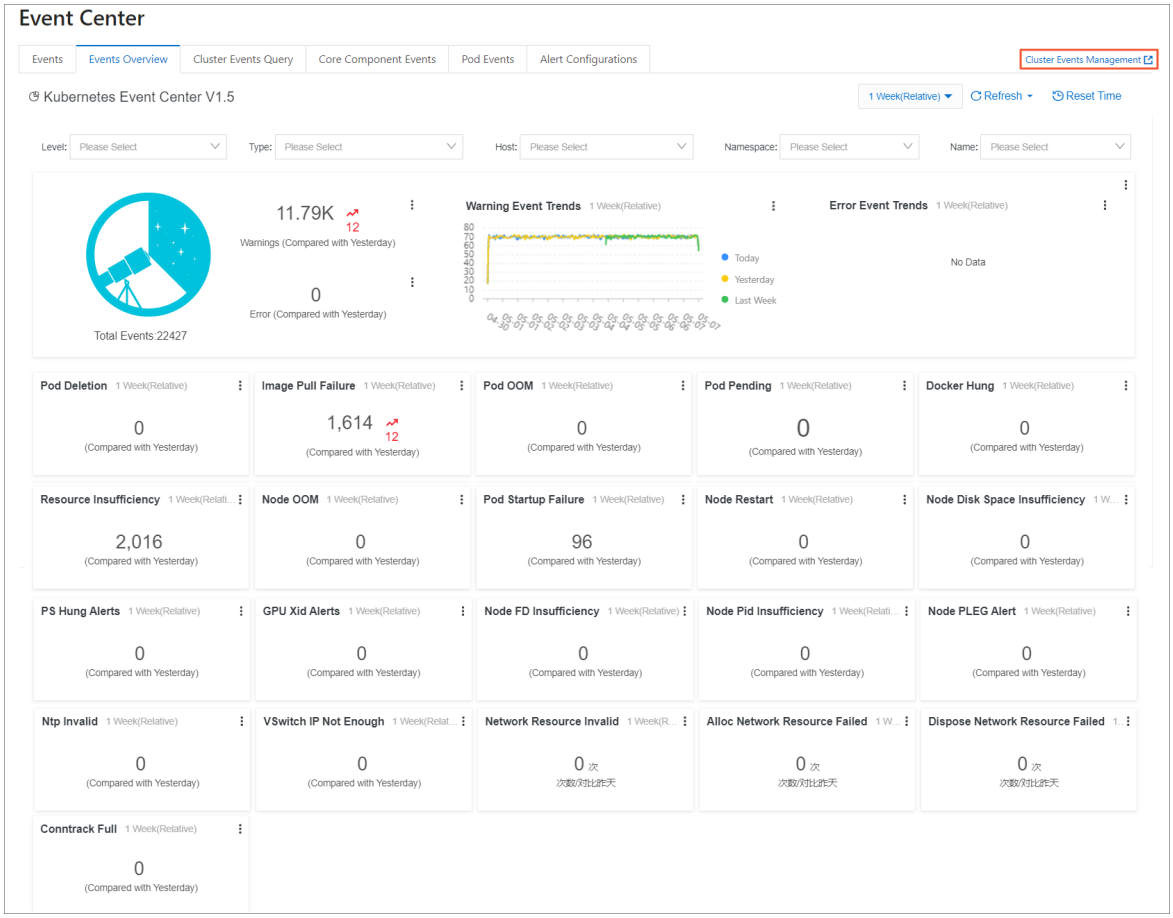
   The system automatically uses the AccessKey pair to access cloud resources when you install the components of the event center.
   Run the following command to create the Secret. The Logtail component uses the Secret.

   ```
   kubectl -n kube-system create secret generic alibaba-addon-secret --from-literal='access-key-id=<your AccessKey ID>' --from-literal='access-key-secret=<your AccessKey Secret>'
   ```

   > **Note** Replace `<your AccessKey ID>` and `<your AccessKey Secret>` with the AccessKey pair that you obtained.

Step 2: Install the Kubernetes event center

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. Choose **Operations > Event Center**.
5. (Optional)If the **ack-node-problem-detector is not installed** message appears, click **Create Event Center**.
6. In the upper-right corner, click **Cluster Events Management**. In the left-side navigation pane of the **K8s Event Center** page, find the cluster that you want to manage. Then, click the > icon to the left of the cluster name to view details of the Kubernetes event center.
   The Kubernetes event center provides event overviews, event details, and information about pod lifecycles. You can also customize queries and configure alerts.



Result

After the configuration is successful, you can use the Kubernetes event center. For more information, see Create and use a Kubernetes event center.

After the Kubernetes event center is created for the external Kubernetes cluster, you can use the event center to check event overviews, view event details, check pod lifecycles, configure alerts, and customize queries.

Related information

- Overview of registered clusters
- Event monitoring

24.3.3. Set up alerting for an external Kubernetes cluster

Container Service for Kubernetes (ACK) allows you to configure alerts to centrally manage exceptions in the cluster and provides various metrics for different scenarios. You can deploy Custom Resource Definitions (CRDs) in a cluster to configure and manage alert rules. This topic describes how to set up alerting and configure alert rules for a registered external Kubernetes cluster.

Prerequisites

- A cluster registration proxy is created and a self-managed Kubernetes cluster is connected to the cluster registration proxy. For more information, see Register an external Kubernetes cluster.
- A kubectl client is connected to the self-managed cluster. For more information, see Connect to Kubernetes clusters by using kubectl.

Scenarios

ACK allows you to configure and manage alerts in a centralized manner to monitor various scenarios. The alerting feature is commonly used in the following scenarios:

- **Cluster O&M**
  You can configure alerts to detect exceptions in cluster management, storage, networks, and elastic scaling at the earliest opportunity. For example, you can configure and enable **Alert Rule Set for Node Exceptions** to monitor exceptions in all nodes or specific nodes in the cluster. You can configure and enable **Alert Rule Set for Storage Exceptions** to monitor changes and exceptions in cluster storage. You can configure and enable **Alert Rule Set for Network Exceptions** to monitor changes and exceptions in cluster networks. You can configure and enable **Alert Rule Set for O&M Exceptions** to monitor changes and exceptions in cluster management operations.
- **Application development**
  You can configure alerts to detect exceptions and abnormal metrics of running applications in the cluster at the earliest opportunity. For example, you can configure alerts to detect exceptions of pod replicas and check whether the CPU and memory usage of a Deployment exceed the thresholds. You can use the default alert template to quickly set up alerts to receive notifications about exceptions of pod replicas in the cluster. For example, you can configure and enable **Alert Rule Set for Pod Exceptions** to monitor exceptions in the pods of your application.
- **Application management**
  To monitor the issues that occur throughout the lifecycle of an application, we recommend that you pay attention to application health, capacity planning, cluster stability, exceptions, and errors. You can configure and enable **Alert Rule Set for Critical Events** to monitor warnings and errors in the cluster. You can configure and enable **Alert Rule Set for Resource Exceptions** to monitor resource usage in the cluster and optimize capacity planning.
- **Multi-cluster management**
  When you manage multiple clusters, you may find it a complex task to configure and synchronize alert rules across the clusters. ACK allows you to **deploy CRDs in the cluster to manage alert rules**. You can configure the same CRDs to conveniently synchronize alert rules across multiple clusters.

Configure the cloud monitoring component in the registered external cluster

Step 1: Configure RAM permissions for the component

Before you can install the component in an external cluster, you must set the AccessKey pair to grant the external cluster the permissions to access Alibaba Cloud resources. Before you set the AccessKey pair, create a Resource Access Management (RAM) user and grant the RAM user the permissions to access Alibaba Cloud resources.

1. Create a RAM user. For more information, see Create a RAM user.
2. Create a permission policy. For more information, see Create a custom policy.

   Use the following template to create a custom permission policy:

```
{
      "Action": [
        "log:*",
        "arms:*",
        "cs:UpdateContactGroup"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
}
```

3. Grant permissions to the RAM user. For more information, see Grant permissions to a RAM user.
4. Create an AccessKey pair for the RAM user. For more information, see Obtain an AccessKey pair.
5. Use the AccessKey pair to create a Secret named alibaba-addon-secret in the self-managed cluster.

   The system automatically uses the AccessKey pair to access cloud resources when you install the cloud monitoring component.

   ```
   kubectl -n kube-system create secret generic alibaba-addon-secret --from-literal='access-key-id=<your access key id>' --from-literal='access-key-secret=<your access key secret>'
   ```

   Note Replace `<your access key id>` and `<your access key secret>` with the AccessKey pair that you obtained.

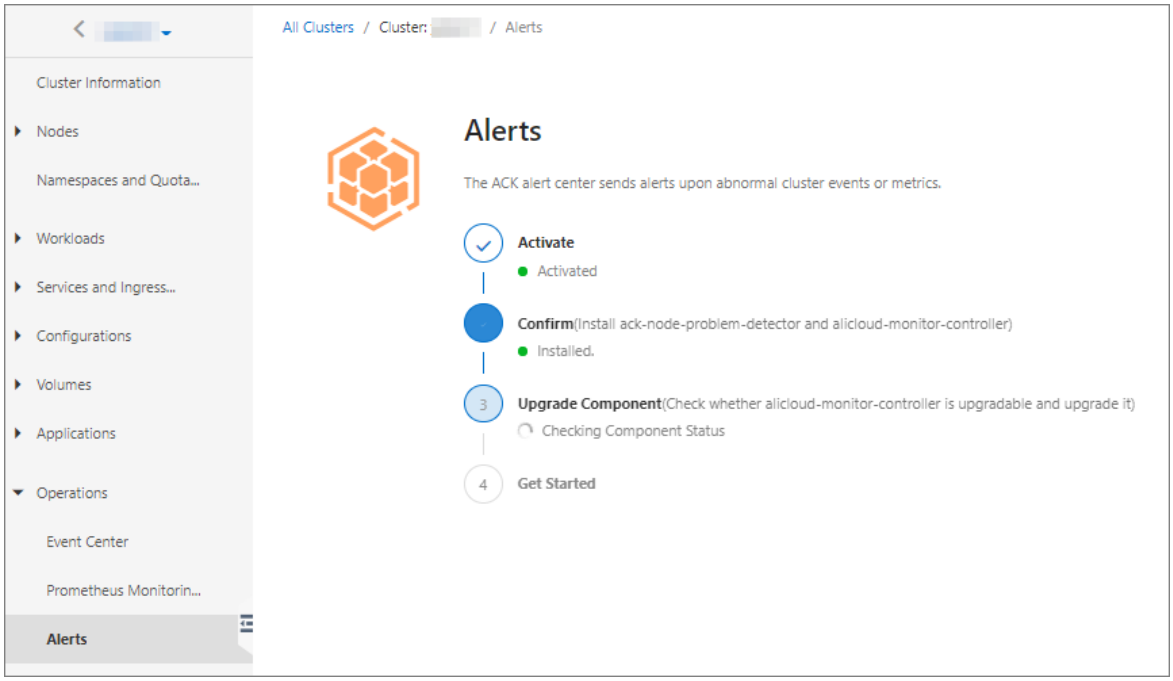
Step 2: Install and upgrade the component

The console automatically checks whether the alerting configuration meets the requirements and guides you to activate, install, or upgrade the component.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Operations > Alerts**.
5. On the **Alerts** page, the console automatically checks whether the following conditions are met.

   If not all conditions are met, follow the on-screen instructions to install or upgrade the required components.

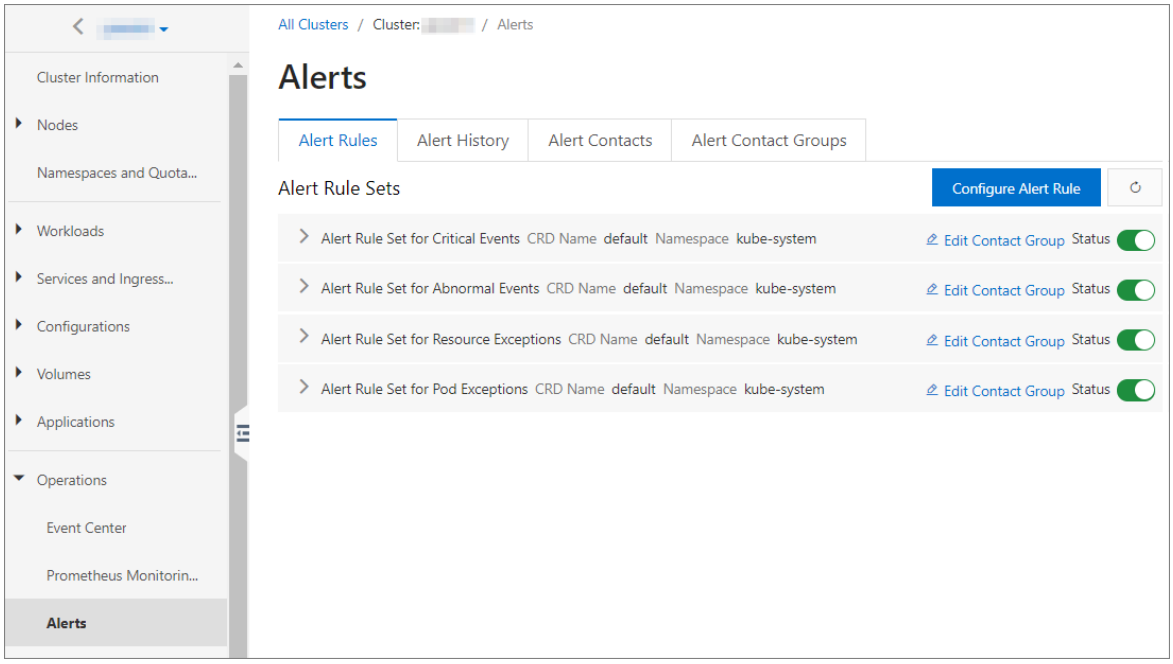
   - Log Service is activated. If you have not activated Log Service, log on to the Log Service console and follow the on-screen instructions to activate the service.
   - **Event Center** is installed. For more information, see Event monitoring.
   - The alicloud-monitor-controller component is upgraded to the latest version. For more information, see alicloud-monitor-controller.



Set up alerting

Step 1: Enable the default alert rules

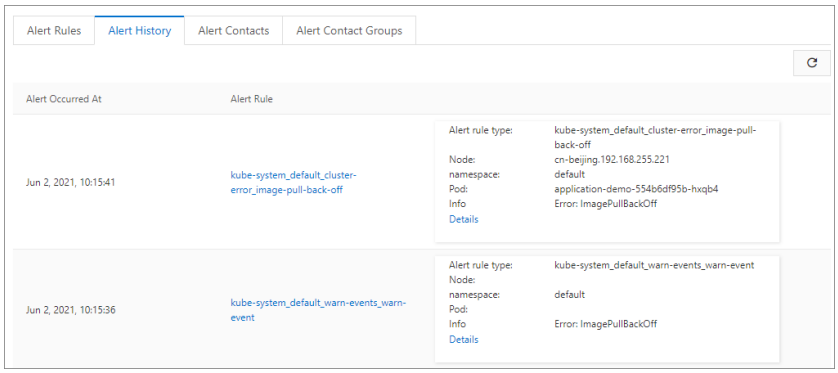
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Operations > Alerts**.
5. On the **Alert Rules** tab, enable the alert rule set.



Step 2: Configure alert rules

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Operations > Alerts**.

Feature	Description

Feature	Description
Alert Rules	○ By default, ACK provides an alert template that is used to generate alerts for exceptions and metrics. ○ Alert rules are classified into several alert rule sets. You can configure multiple alert contact groups for each alert rule set and enable or disable alert rule sets. ○ An alert rule set consists of multiple alert rules, and each alert rule corresponds to an alert item. You can create a YAML file to configure multiple alert rule sets on a cluster. You can also update the YAML file to generate alert rules. ○ For more information about how to configure alert rules by using a YAML file, see Configure alert rules by using CRDs. ○ For more information about the default alert template, see The default alert template.
Alert History	You can view up to 100 historical alerts. You can select an alert and click the link in the **Alert Rule** column to view rule details in the monitoring system. You can click **Details** to go to the resource page where the alert is triggered. The alert may be triggered by an exception or an abnormal metric. 
Alert Contacts	You can create, edit, or delete alert contacts.
Alert Contact Groups	You can create, edit, or delete alert contact groups. When no alert contact group exists, the console automatically creates the default alert contact group based on your registration information.

5. On the **Alert Rules** tab, click **Modify Contacts** to configure the alert contact group to which the alerts are sent. You can turn on or turn off **Status** to enable or disable the alert rule set.

Configure alert rules by using CRDs

When the alerting feature is enabled, the system automatically creates a resource object of the AckAlertRule type in the kube-system namespace. This resource object contains the default alert template. You can use this resource object to configure alert rule sets.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane, choose **Operations > Alerts**.

5. On the **Alert Rules** tab, click **Configure Alert Rule** in the upper-right corner. You can view the configuration of the AckAlertRule resource object and modify the YAML file to update the configuration.

   The following YAML file is provided as an example of the alert rule configuration:

```
apiVersion: alert.alibabacloud.com/v1beta1
kind: AckAlertRule
metadata:
 name: default
spec:
 groups:
  - name: pod-exceptions              ## The name of the alert rule set.
    rules:
     - name: pod-oom                  ## The name of the alert rule.
       type: event                    ## The type of the alert rule. Valid enumeration values: event and metric.
       expression: sls.app.ack.pod.oom     ## The expression of the alert rule. When the type of the alert rule is s
et to event, the expression must be set to sls_event_id, which is the event ID in Log Service.
       enable: enable                 ## The status of the alert rule. Valid enumeration values: enable and disable.
     - name: pod-failed
       type: event
       expression: sls.app.ack.pod.failed
       enable: enable
```

The default alert template

ACK creates the default alert rules on the following conditions:

- The default alert rules are enabled.
- You go to the Alert Rules tab for the first time and default alert rules are not enabled.

The following table describes the default alert rules.

Alert Rule Set	Alert Rule	ACK_CR_Rule_Name	SLS_Event_ID
Alert Rule Set for Critical Events	Errors	error-event	sls.app.ack.error
	Warnings	warn-event	sls.app.ack.warn
Alert Rule Set for Node Exceptions	Docker process exceptions on nodes	docker-hang	sls.app.ack.docker.hang
	Evictions	eviction-event	sls.app.ack.eviction
	GPU XID errors	gpu-xid-error	sls.app.ack.gpu.xid_error
	Node restarts	node-restart	sls.app.ack.node.restart
	Network Time Protocol (NTP) service failures on nodes	node-ntp-down	sls.app.ack.ntp.down
	Pod Lifecycle Event Generator (PLEG) errors on nodes	node-pleg-error	sls.app.ack.node.pleg_error
	Process errors on nodes	ps-hang	sls.app.ack.ps.hang

Alert Rule Set	Alert Rule	ACK_CR_Rule_Name	SLS_Event_ID
Alert Rule Set for Resource Exceptions	Excess file handles on nodes	node-fd-pressure	sls.app.ack.node.fd_pressure
	Insufficient node disk space	node-disk-pressure	sls.app.ack.node.disk_pressure
	Excessive processes on nodes	node-pid-pressure	sls.app.ack.node.pid_pressure
	Insufficient node resources for scheduling	node-res-insufficient	sls.app.ack.resource.insufficient
	Insufficient node IP addresses	node-ip-pressure	sls.app.ack.ip.not_enough
Alert Rule Set for Pod Exceptions	Pod out-of-memory (OOM) errors	pod-oom	sls.app.ack.pod.oom
	Pod restart failures	pod-failed	sls.app.ack.pod.failed
	Image pull failures	image-pull-back-off	sls.app.ack.image.pull_back_off
Alert Rule Set for O&M Exceptions	No available Server Load Balancer (SLB) instance	slb-no-ava	sls.app.ack.ccm.no_ava_slb
	SLB instance update failures	slb-sync-err	sls.app.ack.ccm.sync_slb_failed
	SLB instance deletion failures	slb-del-err	sls.app.ack.ccm.del_slb_failed
	Node deletion failures	node-del-err	sls.app.ack.ccm.del_node_failed
	Node addition failures	node-add-err	sls.app.ack.ccm.add_node_failed
	Route creation failures	route-create-err	sls.app.ack.ccm.create_route_failed
	Route update failures	route-sync-err	sls.app.ack.ccm.sync_route_failed
	High-risk configurations detected in inspections	si-c-a-risk	sls.app.ack.si.config_audit_high_risk
	Command execution failures in managed node pools	nlc-run-cmd-err	sls.app.ack.nlc.run_command_fail
	No command provided in managed node pools	nlc-empty-cmd	sls.app.ack.nlc.empty_task_cmd

Alert Rule Set	Alert Rule	ACK_CR_Rule_Name	SLS_Event_ID
	URL mode not implemented in managed node pools	nlc-url-m-unimp	sls.app.ack.nlc.url_mode_unimpl
	Unknown repair operations in managed node pools	nlc-opt-no-found	sls.app.ack.nlc.op_not_found
	Node draining and removal failures in managed node pools	nlc-des-node-err	sls.app.ack.nlc.destroy_node_fail
	Node draining failures in managed node pools	nlc-drain-node-err	sls.app.ack.nlc.drain_node_fail
	Elastic Compute Service (ECS) restart timeouts in managed node pools	nlc-restart-ecs-wait	sls.app.ack.nlc.restart_ecs_wait_fail
	ECS restart failures in managed node pools	nlc-restart-ecs-err	sls.app.ack.nlc.restart_ecs_fail
	ECS reset failures in managed node pools	nlc-reset-ecs-err	sls.app.ack.nlc.reset_ecs_fail
	Auto-repair task failures in managed node pools	nlc-sel-repair-err	sls.app.ack.nlc.repair_fail
Alert Rule Set for Network Exceptions	Invalid Terway resources	terway-invalid-res	sls.app.ack.terway.invalid_resource
	IP allocation failures of Terway	terway-alloc-ip-err	sls.app.ack.terway.alloc_ip_fail
	Ingress bandwidth configuration parsing failures	terway-parse-err	sls.app.ack.terway.parse_fail
	Network resource allocation failures of Terway	terway-alloc-res-err	sls.app.ack.terway.allocate_failure
	Network resource reclaim failures of Terway	terway-dispose-err	sls.app.ack.terway.dispose_failure
	Terway virtual mode changes	terway-virt-mod-err	sls.app.ack.terway.virtual_mode_change
	Pod IP checks executed by Terway	terway-ip-check	sls.app.ack.terway.config_check
	Ingress configuration reload failures	ingress-reload-err	sls.app.ack.ingress.err_reload_nginx

Alert Rule Set	Alert Rule	ACK_CR_Rule_Name	SLS_Event_ID
Alert Rule Set for Storage Exceptions	Disk size is less than 20 GiB	csi_invalid_size	sls.app.ack.csi.invalid_disk_size
	Subscription disks cannot be mounted	csi_not_portable	sls.app.ack.csi.disk_not_portable
	Unmount failures occur because the mount target is in use	csi_device_busy	sls.app.ack.csi.deivce_busy
	No disks are available	csi_no_ava_disk	sls.app.ack.csi.no_ava_disk
	I/O hangs of cloud disks	csi_disk_iohang	sls.app.ack.csi.disk_iohang
	Slow I/O of the underlying disks of persistent volume claims (PVCs)	csi_latency_high	sls.app.ack.csi.latency_too_high
	Disk usage exceeds the threshold	disk_space_press	sls.app.ack.csi.no_enough_disk_space

24.3.4. Enable ARMS for an external Kubernetes cluster

Application Real-Time Monitoring Service (ARMS) provides a unified method to manage Kubernetes clusters that are deployed across regions. This topic describes how to enable ARMS for an external Kubernetes cluster in the Container Service for Kubernetes (ACK) console.

Prerequisites

An external Kubernetes cluster is registered in the ACK console. For more information, see Register an external Kubernetes cluster.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab, and find and click **ack-arms-pilot**.

   In the upper-right corner of the **App Catalog** page, you can enter **ack-arms-pilot** into the Name search bar and click the search icon. You can also enter a keyword to perform a fuzzy match.
4. On the **App Catalog - ack-arms-pilot** page, select an external cluster in the **Deploy** section to deploy the application.
5. On the **App Catalog - ack-arms-pilot** page, click the **Parameters** tab, set the parameters, and then click **Create** in the **Deploy** section.

Parameter	Description
accessKey	The AccessKey ID of your Alibaba Cloud account. Your account must be authorized to access ARMS.
accessKeySecret	The AccessKey secret of your Alibaba Cloud account.

Notice

- If a leased line is deployed between the cluster and your virtual private cloud (VPC), the leased line is automatically used.
- If the external cluster is registered through a public network, you must delete vpc in the registry address of the image on the Parameters tab.

What's next

Check whether the deployment is successful. For more information, see Install the ARMS agent for a Java application deployed in Container Service for Kubernetes.

Related information

For information about how to use ARMS, see ARMS overview.
Overview of registered clusters

24.3.5. Enable ARMS Prometheus for an external Kubernetes cluster

This topic describes how to enable Application Real-Time Monitoring Service (ARMS) Prometheus for an external Kubernetes cluster by deploying an application in the cluster. This provides a unified approach to manage Kubernetes clusters that are deployed across regions.

Prerequisites

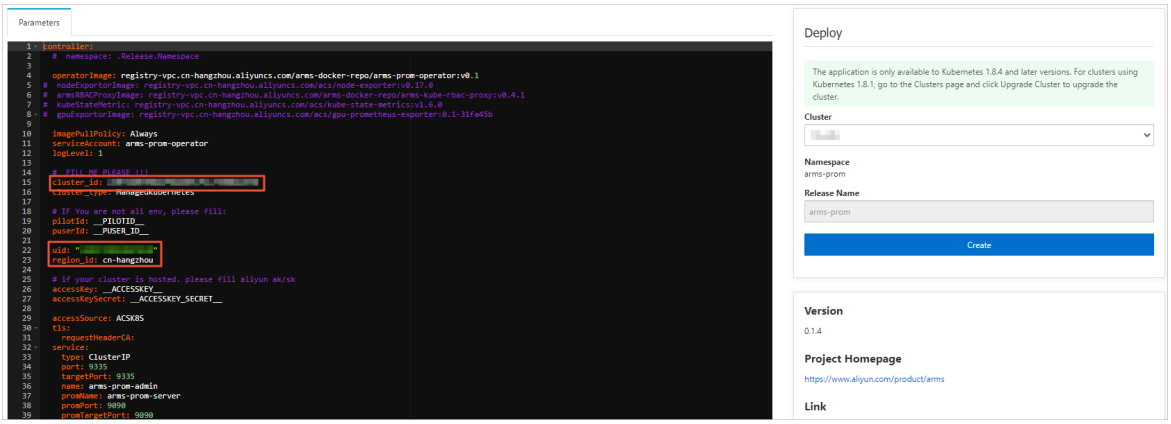
An external Kubernetes cluster is registered in the ACK console. For more information, see Register an external Kubernetes cluster.

Procedure

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab and find and click **ack-arms-prometheus**.

In the upper-right corner of the **App Catalog** page, you can enter **ack-arms-prometheus** in to the Name search bar and click the search icon. You can also enter a keyword to perform a fuzzy match.

4. On the **App Catalog - ack-arms-prometheus** page, select an external cluster to deploy the application and click **Create** in the **Deploy** section.



Parameter	Description	Remarks
cluster_id	The ID of the selected cluster.	These parameters are automatically specified based on the selected cluster.
uid	The user ID of your Alibaba Cloud account. **Note** The value must be enclosed in quotation marks.	
region_id	The region where ARMS is deployed.	

Notice

- If a leased line is deployed between the cluster and virtual private cloud (VPC) where the cluster is deployed, the leased line is automatically used.
- If the external cluster is registered through a public network, you must delete vpc in the address of the image registry on the Parameters tab. For example, after vpc is deleted, the address of the image registry is registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/arms-prom-operator:v0.1.

Result

After the deployment is complete, you can view monitoring data and customize alert rules. For more information, see Enable ARMS Prometheus and Create a Prometheus monitoring alert.

Related information

- Overview of registered clusters

24.3.6. Deploy alibaba-cloud-metrics-adapter in an external Kubernetes cluster

This topic describes how to deploy alibaba-cloud-metrics-adapter in an external Kubernetes cluster in the Container Service for Kubernetes (ACK) console. The alibaba-cloud-metrics-adapter component collects metrics from your cluster. Horizontal Pod Autoscaler (HPA) can scale your application pods based on the collected metrics.

Prerequisites

An external Kubernetes cluster is registered in the ACK console. For more information, see Register an external Kubernetes cluster.

Context

In Kubernetes, metrics are collected to monitor resource usage and performance. In addition, HPA scales the number of pods based on the collected metrics. To meet diverse monitoring requirements of developers, Kubernetes defines the following APIs: resource metrics, custom metrics, and external metrics.

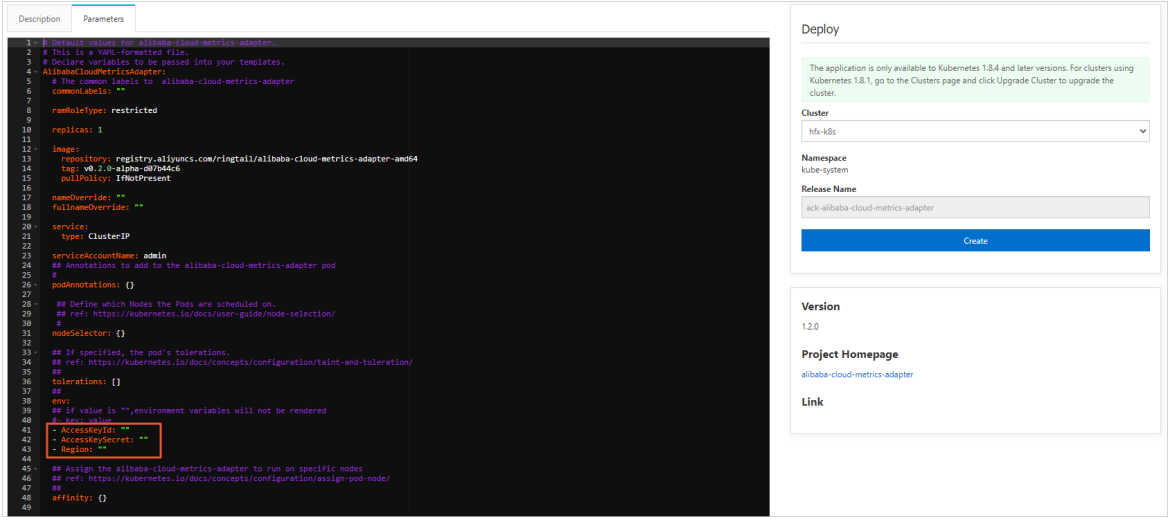
- Resource metrics are collected by the metrics-server component. The collected metrics are used to monitor the usage of Kubernetes resources, such as pods, nodes, and namespaces.
- Custom metrics are collected by ARMS Prometheus. HPA scales application pods based on the collected custom metrics.
- External metrics are collected through the external metrics API provided by the cloud service provider. These metrics are used to monitor the external environment. For example, you can monitor the queries per second (QPS) to an Ingress and use HPA to scale application pods when the QPS exceeds the scaling threshold.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click the **Alibaba Cloud Apps** tab and find **ack-alibaba-cloud-metrics-adapter**.

   In the upper-right corner of the **App Catalog** page, you can enter **ack-alibaba-cloud-metrics-adapter** into the Name search bar and click the search icon. You can also enter a keyword to perform a fuzzy match.
4. On the **App Catalog - ack-alibaba-cloud-metrics-adapter** page, select a cluster in the **Deploy** section to deploy the application.

   Use the default values for the **Namespace** and **Release Name** parameters.
5. On the **App Catalog - ack-alibaba-cloud-metrics-adapter** page, click the **Parameters** tab, set the parameters, and click **Create** in the **Deploy** section.



Parameter	Description
AccessKeyId	The AccessKey ID of your Alibaba Cloud account.
AccessKeySecret	The AccessKey secret of your Alibaba Cloud account.
Region	The region where your cluster is deployed, for example, cn-qingdao or ap-southeast-1.

> **Note** If a leased line is deployed between the cluster and your virtual private cloud (VPC), the leased line is automatically used.

Related information

- Overview of registered clusters
-

24.3.7. Use the inspection feature to check for security risks in the workloads of an ACK cluster

This topic describes how to use the inspection feature to check for security risks in the workloads of a Container Service for Kubernetes (ACK) cluster. This topic also describes how to view inspection reports. This way, you can detect potential risks in workloads at the earliest opportunity.

Prerequisites

An external Kubernetes cluster is registered in the Container Service for Kubernetes (ACK) console. For more information, see Register an external Kubernetes cluster.

Grant permissions to a RAM user

If you log on as a Resource Access Management (RAM) user, you must authorize the RAM user to access the specified Log Service project. Otherwise, you cannot access the specified Log Service project. For more information, see Use custom policies to grant permissions to a RAM user.

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "log:Get*",
        "log:List*"
      ],
      "Resource": "acs:log:*:*:project/<The name of the project>/*",
      "Effect": "Allow"
    }
  ]
}
```

Inspect workloads in an ACK cluster

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or

click **Details** in the **Actions** column. The details page of the cluster appears.

4. In the left-side navigation page of the cluster details page, choose **Security > Inspections**.
5. If the inspection component is not installed, click **Install** below **Confirm**. If the inspection component is installed, skip this step.
6. In the upper-right corner of the **Inspections** page, click **Inspect**.
7. After the inspection is completed, click the Refresh button to view the inspection report.
8. (Optional)In the upper-right corner of the **Inspections** page, click **Configure Periodic Inspection**. In the panel that appears, you can enable or disable periodic inspections and configure inspection items.

Inspection details

The **Inspections** page provides a table to show the inspection results of different workloads. The following features are provided to display the inspection results:

- Displays the values of **Number of Passed Items** and **Number of Failed Items** for each inspected workload.
- Displays the passed and failed inspection items, description of each inspection item, and suggestions for security reinforcement on the inspection details page.
- Allows you to configure workload whitelists for inspection.
- Allows you to select filter options from the **Namespace**, **Workload Type**, and **Passed or Failed** drop-down lists to narrow down inspection results.

Inspection reports

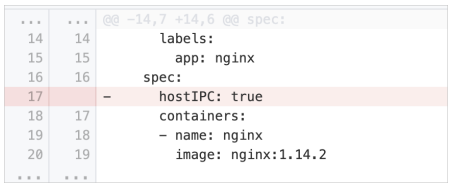
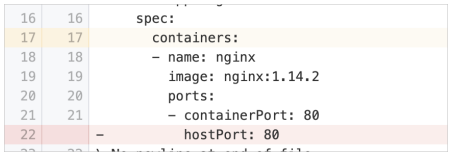
An inspection report provides the results of the latest inspection, including the following information:

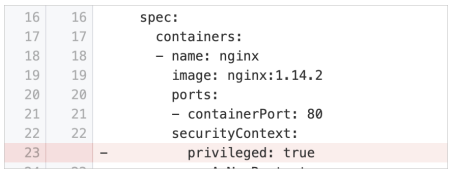
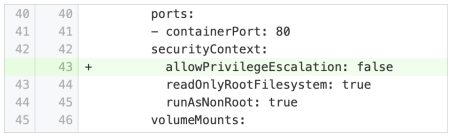
- Overview of the inspection results. This includes the total number of inspection items, the number and percentage of each inspected resource object type, and the overall health status of the cluster.
- Statistics of the following inspection categories: health checks, images, networks, resources, and security conditions.
- Detailed inspection results of the configurations of each workload. The results include resource categories, resource names, namespaces, inspection types, inspection items, and inspection results.

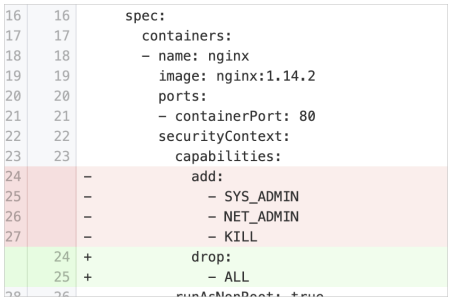
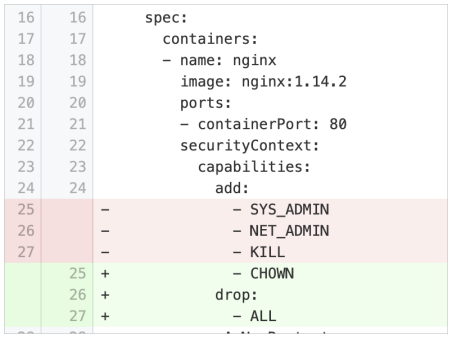
The following table describes the inspection items.

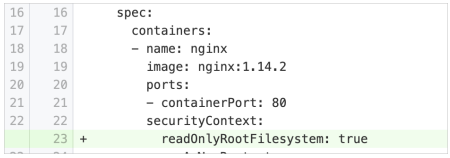
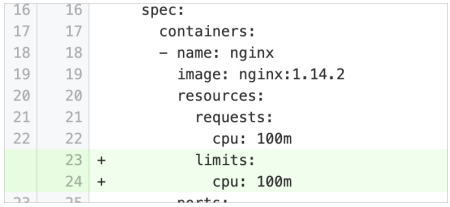
Inspection item	Inspection content and potential security risk	Suggestion

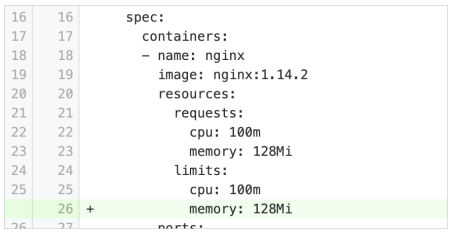
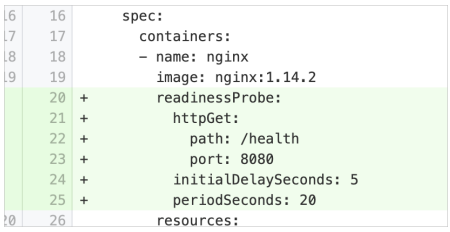
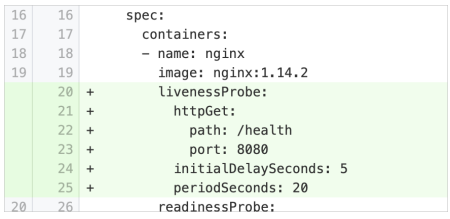
hostNetworkSet	Checks whether the pod specification of a workload contains the hostNetwork:true setting. This setting specifies that the pod uses the network namespace of the host. If the hostNetwork:true setting is specified, the host network may be attacked by containers in the pod and the data transfer in the host network may be sniffed.	Delete the hostNetwork field from the pod specification.
		Example:
		`@@ -14,7 +14,6 @@ spec:`
		`labels:`
		`app: nginx`
		`spec:`
		`- hostNetwork: true`
		`containers:`
		`- name: nginx`
		`image: nginx:1.14.2`

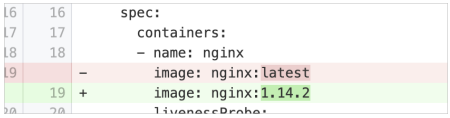
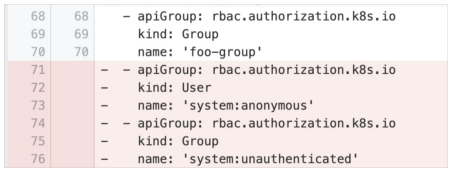
Inspection item	Inspection content and potential security risk	Suggestion
hostIPCSet	Checks whether the pod specification of a workload contains the hostIPC:true setting. This setting specifies that the pod uses the inter-process communication (IPC) namespace of the host. If the hostIPC:true setting is used, containers in the pod may attack the host processes and sniff process data.	Delete the hostIPC field from the pod specification. Example: 
hostPIDSet	Checks whether the pod specification of a workload contains the hostPID:true setting. This setting specifies that the pod uses the process ID (PID) namespace of the host. If the hostPID:true setting is used, containers in the pod may attack the host processes and collect process data.	Delete the hostPID field from the pod specification. Example: 
hostPortSet	Checks whether the pod specification of a workload contains the hostPort field. This field specifies the host port to which the listening port of the pod is mapped. If the hostPort field is specified, the specified host port may be occupied without authorization and the container port may receive unexpected requests.	Delete the hostPort field from the pod specification. Example: 
runAsRootAllowed	Checks whether the pod specification of a workload contains the runAsNonRoot:true setting. This setting specifies that containers in the pod are not allowed to run as the root user. If the runAsNonRoot:true setting is not used, malicious processes in the containers may intrude into your applications, hosts, or cluster.	Add the runAsNonRoot:true setting to the pod specification. Example: 

Inspection item	Inspection content and potential security risk	Suggestion
runAsPrivileged	Checks whether the pod specification of a workload contains the privileged:true setting. This setting specifies that containers in the pod are allowed to run in privileged mode. If the privileged:true setting is used, malicious processes in the containers may intrude into your applications, hosts, or cluster.	Delete the privileged field from the pod specification. Example: 
privilegeEscalationAllowed	Checks whether the pod specification of a workload contains the allowPrivilegeEscalation:false setting. This setting specifies that the child processes of a container cannot be granted higher privileges than the parent process. If the allowPrivilegeEscalation:false setting is not used, malicious processes in the container may be granted escalated privileges and perform unauthorized operations.	Add the allowPrivilegeEscalation:false setting to the pod specification. Example: 

Inspection item	Inspection content and potential security risk	Suggestion
capabilitiesAdded	Checks whether the pod specification of a workload contains the capabilities field. This field is used to enable Linux capabilities for processes in containers. The capabilities include SYS_ADMIN, NET_ADMIN, and ALL. If the capabilities field is specified, malicious processes in the containers may intrude into your applications, cluster components, or cluster.	Modify the pod specification to retain only the required Linux capabilities and remove other capabilities. If processes in the containers do not require Linux capabilities, remove all Linux capabilities. Example 1:  If processes in the containers require Linux capabilities, specify only the required Linux capabilities and remove other capabilities. Example 2: 

Inspection item	Inspection content and potential security risk	Suggestion
notReadOnlyRootFileSystem	Checks whether the pod specification of a workload contains the readOnlyRootFilesystem:true setting. This setting specifies that the root file system mounted to containers is read-only. If the readOnlyRootFilesystem:true setting is not used, malicious processes in the containers may modify the root file system.	Add the readOnlyRootFilesystem:true setting to the pod specification. If you want to modify files in a specific directory, set volumeMounts in the pod specification. Example:  If you want to modify files in a specific directory, set the volumeMounts field in the pod configurations. Example: 
cpuRequestsMissing	Checks whether the pod specification of a workload contains the resources.requests.cpu field. This field specifies the minimum CPU resources that are required to run each container. If the resources.requests.cpu field is not specified, the pod may be scheduled to a node that has insufficient CPU resources. This may lead to slow processes.	Add the resources.requests.cpu field to the pod specification. Example: 
cpuLimitsMissing	Checks whether the pod specification of a workload contains the resources.limits.cpu field. This field specifies the maximum CPU resources that can be used to run each container. If the resources.limits.cpu field is not specified, abnormal processes in containers may consume an excessive amount of CPU resources or exhaust the CPU resources of the node or cluster.	Add the resources.limits.cpu field to the pod specification. Example: 

Inspection item	Inspection content and potential security risk	Suggestion
memoryRequestsMissing	Checks whether the pod specification of a workload contains the resources.requests.memory field. This field specifies the minimum memory resources that are required to run each container. If the resources.requests.memory field is not specified, the pod may be scheduled to a node that has insufficient memory resources. As a result, processes in containers may be terminated when the Out of Memory (OOM) killer is triggered.	Add the resources.requests.memory field to the pod specification. Example: 
memoryLimitsMissing	Checks whether the pod specification of a workload contains the resources.limits.memory field. This field specifies the maximum memory resources that can be used to by each container. If the resources.limits.memory field is not specified, abnormal processes in containers may consume an excessive amount of memory resources or exhaust the memory resources of the node or cluster.	Add the resources.limits.memory field to the pod specification. Example: 
readinessProbeMissing	Checks whether the pod specification of a workload contains the readinessProbe field. This field specifies whether readiness probes are configured for containers. Readiness probes are used to check whether applications in the containers are ready to process requests. If the readinessProbe field is not specified, service exceptions may occur when requests are sent to applications that are not ready to process requests.	Add the readinessProbe field to the pod specification. Example: 
livenessProbeMissing	Checks whether the pod specification of a workload contains the livenessProbe field. This field specifies whether liveness probes are configured for containers. Liveness probes are used to check whether a container restart is required to resolve application exceptions. If the livenessProbe field is not specified, the service may be interrupted when application exceptions can be resolved only by restarting containers.	Add the livenessProbe field to the pod specification. Example: 

Inspection item	Inspection content and potential security risk	Suggestion
tagNotSpecified	Checks whether the image field in the pod specification of a workload specifies an image version or whether the value of the field is set to latest. If no image version is specified or the value of the field is set to latest, the service may be interrupted when containers use a wrong image version.	Modify the image field in the pod specification by specifying an image version. Set the field to a value other than latest. Example: 
anonymousUserRBACBinding	Checks role-based access control (RBAC) role bindings in the cluster and locates the configurations that allow access from anonymous users. If anonymous users are allowed to access the cluster, they may gain access to sensitive information, attack the cluster, and intrude into the cluster.	Remove the configurations that allow access from anonymous users from the RBAC role bindings. Example: 

Related information

- Configure a Security Context for a Pod or Container
- Configure Liveness, Readiness and Startup Probes

24.4. FAQ about multi-cloud and hybrid cloud

This topic provides answers to some frequently asked questions about multi-cloud and hybrid cloud.

Are registered Kubernetes clusters free of charge?

Registered Kubernetes clusters are free of charge. Fees are charged when cloud resources are used. For more information, see Resources that are involved when you register clusters.

Can I use cloud computing resources to scale out an external Kubernetes cluster that is deployed in a data center?

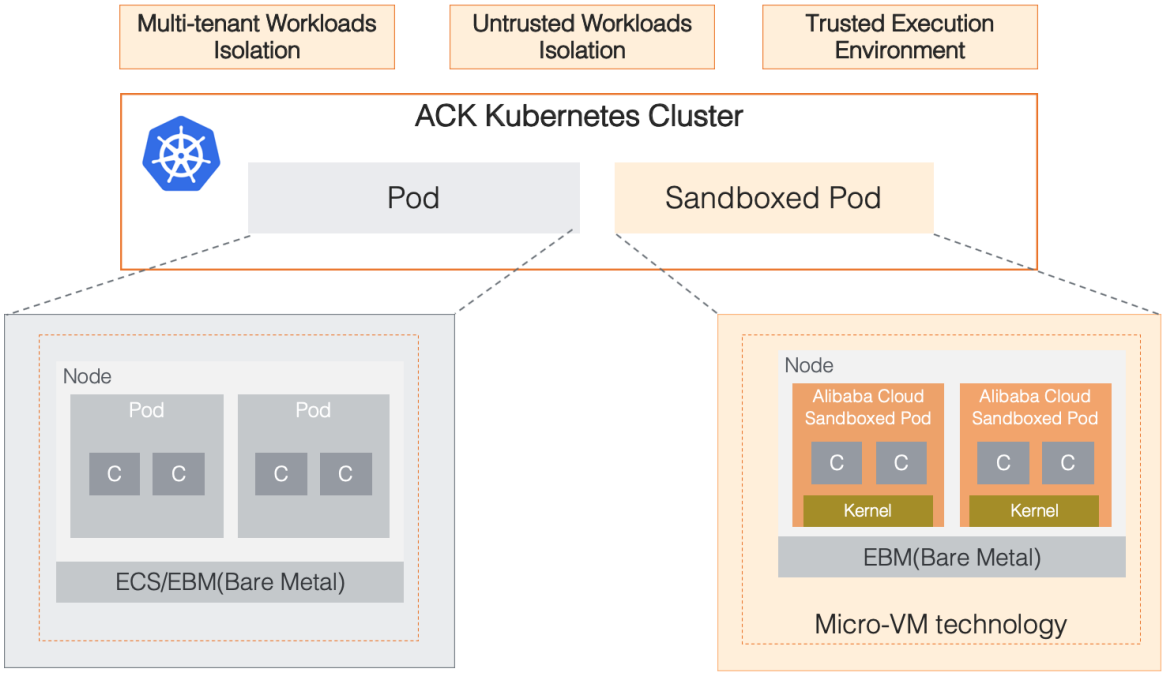
Yes. For more information, see Create a hybrid cluster.

25.Sandboxed-Container management

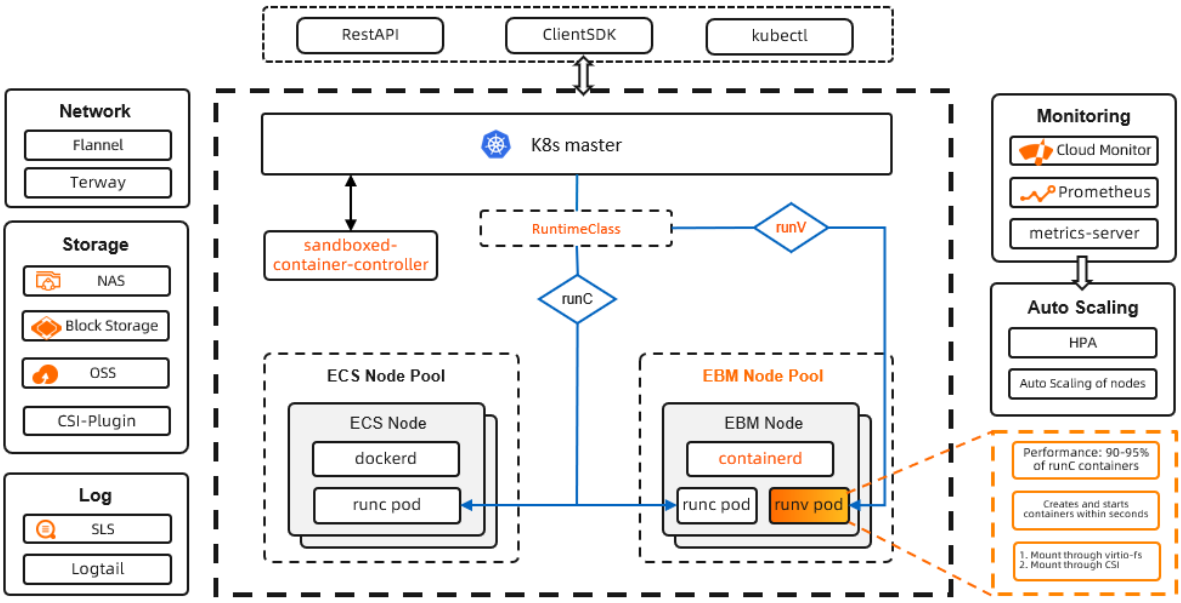
25.1. Sandboxed-Container overview

Sandboxed-Container provides an alternative to the Docker runtime. It allows you to run applications in a sandboxed and lightweight virtual machine that has a dedicated kernel. This enhances resource isolation and improves security.

Sandboxed-Container is suitable in scenarios such as untrusted application isolation, fault isolation, performance isolation, and load isolation among multiple users. Sandboxed-Container provides higher security. Sandboxed-Container has minor impacts on application performance and offers the same user experience as Docker in terms of logging, monitoring, and elastic scaling.



Architecture



Features

Sandboxed-Container is container-securing runtime that is developed by Alibaba Cloud based on sandboxed and lightweight virtual machines. Compared with Sandboxed-Container V1, Sandboxed-Container V2 maintains the same isolation performance and reduces the pod overhead by 90%. It also allows you to start sandboxed containers 3 times faster and increases the maximum number of pods that can be deployed on a host by 10 times. Sandboxed-Container V2 provides the following key features:

- Strong isolation based on sandboxed and lightweight virtual machines.
- Good compatibility with runC in terms of application management.
- High performance that corresponds to 90% the performance of applications based on runC.
- Network Attached Storage (NAS) file systems, Alibaba Cloud disks, and OSS buckets can be mounted both directly and through virtio-fs.
- The same user experience as that provided by containers in runC in terms of monitoring, logging, and storage.
- Support for RuntimeClass (runC and runV). For more information, see RuntimeClass.
- Less requirements on technical expertise and skills of using virtual machines.
- Higher stability than that provided by Kata Containers. For more information about Kata Containers, see Kata Containers.

Related information

- Comparison of Docker, containerd, and Sandboxed-Container
- Differences between runC and runV
- Benefits of Sandboxed-Container
- Create a managed Kubernetes cluster that runs sandboxed containers

25.2. Differences between runC and runV

This topic describes the differences between runC and Sandboxed-Container (runV) in terms of their performance and pod creation methods. This allows you to better understand and utilize the benefits of sandboxed containers.

Differences between runC and runV

Item	runC	runV
Container engine	Docker and Containerd	Containerd
Node type	Elastic Compute Service (ECS) instances and ECS Bare Metal instances	EBM
Container kernel	Share the host kernel	Dedicated kernel
Container isolation	Cgroups and namespaces	Lightweight virtual machines (VMs)
Rootfs Graph Driver	OverlayFS	DeviceMapper
RootFS I/O throttling	Cgroups	DeviceMapper Block IO Limit **Note** Supported by only Sandboxed-Container V1.
NAS mounting	Not supported	Supported
Disk mounting	Not supported	Supported
Container log collection	Logtail directly collects container logs from the host.	Logtail sidecar. For more information, see 通过Sidecar-CRD方式采集容器日志.
Pod Overhead	None	• Sandboxed-Container V1: For example, if you set memory: 512 Mi for a pod overhead, it indicates that 512 MiB of memory is allocated to the pod sandbox. In this case, if you set a memory limit of 512 MiB for containers in the pod, the pod will request a total memory of 1,024 MiB. • Sandboxed-Container V2: The memory limit for a pod overhead is calculated based on the formula: Memory for a pod overhead = 64 MiB + requested memory of containers in a pod × 2%. If the result is more than 512 MiB, the value is set to 512 Mi. If the result is less than 64 MiB, the value is set to 64 Mi.

Differences in pod creation between runC and runV

You can connect to clusters of Alibaba Cloud Container Service for Kubernetes (ACK) by using the kubectl command-line tool. For more information, see Connect to Kubernetes clusters by using kubectl.

- Create a pod that uses runC
  i. (Optional)Use `runtimeClassName: runc` to set the container runtime to runC.

  Note The preceding command is optional. runC is the default container runtime.

  ii. Run the following command to create a pod that uses runC:

```
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Pod
metadata:
 name: busybox-runc
 labels:
  app: busybox-runc
spec:
 containers:
 - name: busybox
  image: registry.cn-hangzhou.aliyuncs.com/acs/busybox:v1.29.2
  command:
  - tail
  - -f
  - /dev/null
  resources:
   limits:
    cpu: 1000m
    memory: 512Mi
   requests:
    cpu: 1000m
    memory: 512Mi
EOF
```

- Create a pod that uses runV
  i. Use `runtimeClassName: runv` to set the container runtime to runV.
  ii. (Optional)Run the following command to verify that a RuntimeClass object named `runv` exists in the cluster.

  ```
  kubectl get runtimeclass runv -o yaml
  ```

  > **Note** A RuntimeClass object named `runv` is automatically created in a Kubernetes cluster that uses Sandboxed-Container.

  iii. Run the following command to create a pod that uses runV:

```
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Pod
metadata:
 name: busybox-runv
 labels:
  app: busybox-runv
spec:
 runtimeClassName: runv
 nodeSelector:
  alibabacloud.com/container-runtime: Sandboxed-Container.runv
 containers:
 - name: busybox
  image: registry.cn-hangzhou.aliyuncs.com/acs/busybox:v1.29.2
  command:
  - tail
  - -f
  - /dev/null
  resources:
   limits:
    cpu: 1000m
    memory: 512Mi
   requests:
    cpu: 1000m
    memory: 512Mi
EOF
```

Notice If your Kubernetes version is earlier than 1.16, add the following nodeSelector configuration.

```
nodeSelector:
  alibabacloud.com/container-runtime: Sandboxed-Container.runv
```

iv. Run the following command to query the created pod. If the output is `runv`, it indicates that the pod is running in a sandbox.

```
kubectl get pod busybox-runv -o jsonpath={.spec.runtimeClassName}
```

v. Run the following command to log on to the pod and query its CPU and memory specifications.

```
kubectl exec -ti pod busybox-runv /bin/sh
/ # cat /proc/meminfo | head -n1
MemTotal:     1130692 kB
/ # cat /proc/cpuinfo | grep processor
processor   : 0
```

The output shows that the number of CPUs is not the same as that of the host. The total memory is the sum of pod memory and pod overhead. Note that the total memory is slightly smaller because the system uses some memory as well.

Related information

- Compatibility notes
- RuntimeClass

25.3. Comparison of Docker, containerd, and Sandboxed-Container

Containers and images have become industry standards for software packaging and delivery. Kubernetes has become a standard platform for building, developing, and managing containerized cloud-native applications. An increasing number of enterprises and customers choose to deploy their applications in Container Service for Kubernetes (ACK). ACK supports the containerd, Docker, and Sandboxed-Container runtimes. This topic compares these runtimes in terms of implementations, limits, and deployment architectures. It also compares the commonly used commands provided by Docker Engine and containerd. This allows you to select a container runtime based on your requirements and scenarios.

Comparison in terms of implementations and limits

Item	containerd	Docker	Sandboxed-Container	Description
Cluster type	Managed Kubernetes clusters and dedicated Kubernetes clusters	All types	Managed Kubernetes clusters and dedicated Kubernetes clusters	None
Node type	Supports: • ECS • EBM	Supports: • ECS • EBM	Supports: • EBM	None
Node OS	Supports: • CentOS • Alibaba Cloud Linux 2 • ACK v1.20.4 version of Windows	Supports: • CentOS • Alibaba Cloud Linux 2 • Windows	Supports: • Alibaba Cloud Linux 2 Customized Edition	• You cannot deploy both Docker and Sandboxed-Container on a node. • To deploy both Docker and Sandboxed-Container in a cluster, you can create node pools of different runtime types.
Container engine	containerd	Docker Engine	containerd	None
Monitoring	Supported	Supported	Supported	None
Container log collection	Supported	Supported	Supports log collection by using sidecar containers. Manual configuration is required.	For more information about sidecar configurations, see 通过Sidecar-CRD方式采集容器日志.
Container stdout collection	Supported	Supported	Supported	None

Item	containerd	Docker	Sandboxed-Container	Description
RuntimeClass	Not supported	Not supported	Supported (runV)	None
Pod scheduling	No configuration is required.	No configuration is required.	You must add configurations based on the following rules: • For Kubernetes 1.14.x, you must add the following configuration to the nodeSelector field. **alibabacloud.com/sandboxed-container: Sandboxed-Container.runv** • For Kubernetes V1.16.x and later, no extra configuration is required.	None
HostNetwork	Supported	Supported	Not supported	None
exec/logs	Supported	Supported	Supported	None
Node data disk	Optional	Optional	Required. The data disk must be at least 200 GiB.	None
Network plug-in	Supports: • Flannel • Terway	Supports: • Flannel • Terway	Supports: • Flannel • Terway: supports only the inclusive ENI mode.	None
kube-proxy mode	Supports: • Iptables • IPVS	Supports: • Iptables • IPVS	Supports: • Iptables • IPVS	None

Item	containerd	Docker	Sandboxed-Container	Description
Volume plug-in	CSI	CSI	CSI	None
Container root file system	OverlayFS	OverlayFS	DeviceMapper	None

Comparison in terms of deployment architectures

Runtime	Deployment architecture
Docker	kubelet -> dockerd -> containerd -> containerd-shim -> runC containers
Containerd	kubelet -> containerd -> containerd-shim -> runC containers
Sandboxed-Container V1	kubelet -> (CRI)containerd \-> containerd-shim -> runC containers \-> containerd-shim-kata-v2 -> runV sandboxed containers
Sandboxed-Container V2	kubelet -> (CRI)containerd \-> containerd-shim -> runC containers \-> containerd-shim-rund-v2 -> runV sandboxed containers

Comparison of the commonly used commands provided by Docker Engine and containerd

Docker uses Docker Engine for container lifecycle management. Sandboxed-Container uses containerd for container lifecycle management. These tools provide different commands that can be used to manage images and containers. The following table describes the commonly used commands provided by Docker Engine and containerd.

Description	Docker	Containerd	
	docker	crictl (recommended)	ctr
Queries containers.	docker ps	crictl ps	ctr -n k8s.io c ls
Queries information about one or more containers.	docker inspect	crictl inspect	ctr -n k8s.io c info
Queries container logs.	docker logs	crictl logs	N/A
Runs a command in a container.	docker exec	crictl exec	N/A

Description	Docker	Containerd	
	docker	crictl (recommended)	ctr
Attaches to a container.	docker attach	crictl attach	N/A
Queries resource usage statistics.	docker stats	crictl stats	N/A
Creates a container.	docker create	crictl create	ctr -n k8s.io c create
Starts one or more containers.	docker start	crictl start	ctr -n k8s.io run
Stops one or more containers.	docker stop	crictl stop	N/A
Removes one or more containers.	docker rm	crictl rm	ctr -n k8s.io c del
Queries images.	docker images	crictl images	ctr -n k8s.io i ls
Queries information about one or more images.	docker inspect	crictl inspecti	N/A
Pulls an image.	docker pull	crictl pull	ctr -n k8s.io i pull
Pushes an image.	docker push	N/A	ctr -n k8s.io i push
Removes one or more images.	docker rmi	crictl rmi	ctr -n k8s.io i rm
Queries pods.	N/A	crictl pods	N/A
Queries information about one or more pods.	N/A	crictl inspectp	N/A
Starts a new pod.	N/A	crictl runp	N/A
Stops one or more pods	N/A	crictl stopp	N/A

25.4. Benefits of Sandboxed-Container

This topic describes the advantages and application scenarios of Sandboxed-Container and provides a comparison between Sandboxed-Container and open source Kata Containers. This allows you to learn more about the benefits of Sandboxed-Container.

Context

Sandboxed-Container provides an alternative to the Docker runtime environment. It supports the following features:

- Sandboxed-Container allows your applications to run in a sandboxed and lightweight virtual machine. This virtual machine is equipped with a dedicated kernel and provides better isolation and enhanced security.
- Compared with open source Kata Containers, Sandboxed-Container is optimized in terms of storage, networking, and stability.

- You can use Sandboxed-Container to isolate untrusted applications and applications of different tenants for higher security. You can also use Sandboxed-Container to isolate applications with faults and applications with degraded performance. This minimizes the negative impact on your service. In addition, Sandboxed-Container offers the same user experience as Docker in terms of logging, monitoring, and elastic scaling.

Benefits

Compared with Docker, Sandboxed-Container has the following benefits:

- Strong isolation based on sandboxed and lightweight virtual machines.
- Network Attached Storage (NAS) file systems and Alibaba Cloud disks can be mounted to Sandboxed-Container. This provides the same storage performance as storage volumes that are mounted to the host.
- Compatibility with runC in terms of application management.
- High performance that corresponds to 90% performance of applications based on runC.
- The same user experience as runC in terms of logging, monitoring, and storage.
- Support for RuntimeClass.
- Easy to use with limited expertise that is required to use virtual machines.
- Higher stability than that provided by Kata Containers.

Comparison between Sandboxed-Container and Kata Containers

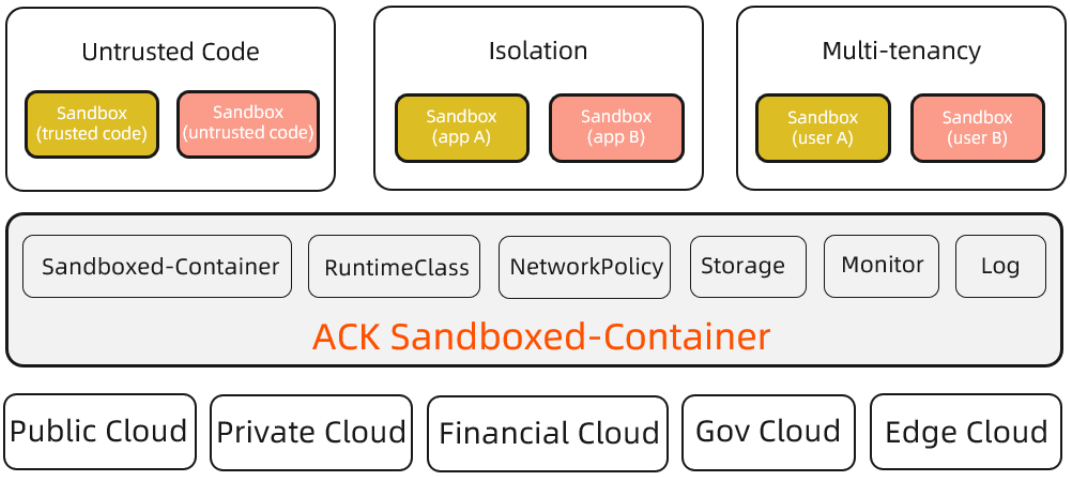
Sandboxed-Container outperforms Kata Containers in the following aspects.

Item	Category	Sandboxed-Container	Kata Containers
Sandbox startup time consumption		About 150 ms	About 500 ms
Root file system		OverlayFS over virtio-fs. Performance: ☆☆☆☆	OverlayFS over 9pfs. Performance: ☆☆
Volume	HostPath	Disks are mounted to Sandboxed-Container over 9pfs. Performance: ☆☆	Disks are mounted to Kata Containers over 9pfs. Performance: ☆☆
	EmptyDir	The volume is mounted to Sandboxed-Container over virtio-fs.	By default, the volume is mounted to Kata Containers over 9pfs.
	Cloud disks	• By default, cloud disks are mounted to Sandboxed-Container over virtio-fs. Performance: ☆☆☆☆ • Cloud disks are mounted to Sandboxed-Container after you set specific container environment variables. Performance: ☆☆☆☆☆	Cloud disks are mounted to Kata Containers over 9pfs. Performance: ☆☆

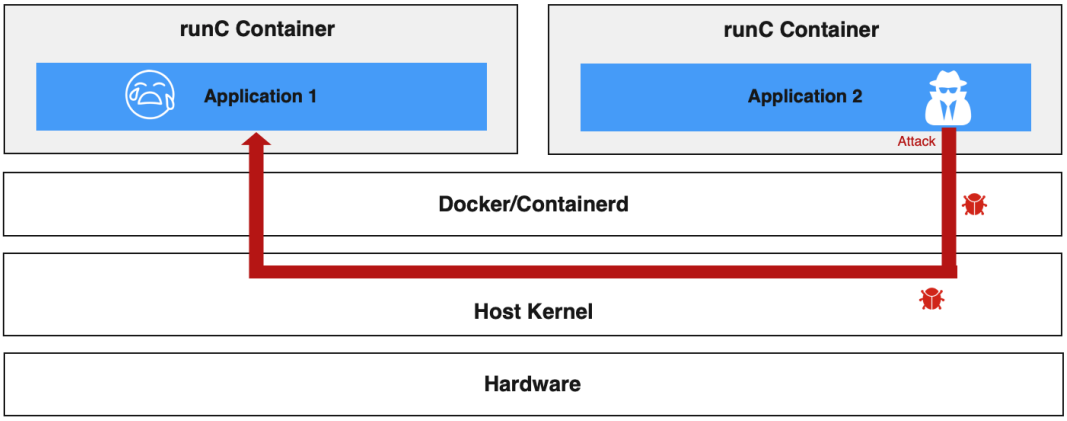
Item	Category	Sandboxed-Container	Kata Containers
	NAS	• By default, Apsara File Storage NAS (NAS) file systems are mounted to Sandboxed-Container over virtio-fs. Performance: ☆☆☆☆ • NAS file systems are mounted to Sandboxed-Container after you set specific container environment variables. Performance: ☆☆☆☆☆	NAS file systems are mounted to Kata Containers over 9pfs. Performance: ☆
Network Plug-in		• The Terway network plug-in is used. Its network performance is 20% to 30% higher than Flannel. Terway supports features such as NetworkPolicy. This allows you to define the networking policies for pods. For more information, see Terway plug-in. • Flannel	Flannel
Monitoring and alerting		• Enhanced monitoring of disks and network conditions for pods that host Sandboxed-Container. • Integrated with Cloud Monitor. This facilitates cluster monitoring and alerting.	Monitoring of disks and network conditions is unavailable for pods that host Sandboxed-Container.
Stability		☆☆☆☆☆	☆☆

Applicable scenarios of Sandboxed-Container

This section describes the applicable scenarios of Sandboxed-Container.



- Scenario 1: Sandboxed-Container can run untrusted code and applications in isolated containers. This is not supported by containers in runC.
  - Security risks of runC



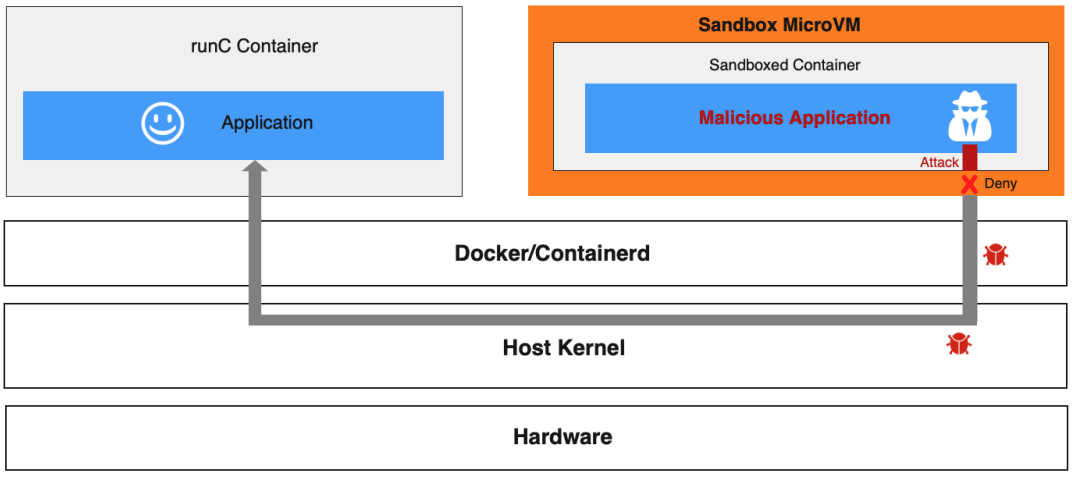
    - runC isolates containers by using namespaces and control groups (cgroups). This exposes containers to security threats.
    - All containers on a node share the host kernel. If a kernel vulnerability is exposed, malicious code may escape to the host and then infiltrate the backend network. Malicious code execution may cause privilege escalation, compromise sensitive data, and destroy system services and other applications.
    - Attackers may also exploit application vulnerabilities to infiltrate the internal network.

    You can implement the following measures to reduce security risks of containers in runC.

    - Seccomp: filters system calls.
    - SElinux: restricts the permissions of container processes, files, and users.
    - Capability: limits the capability of container processes.
    - dockerd rootless mode: forbids users to use root permissions to run the Docker daemon and containers.

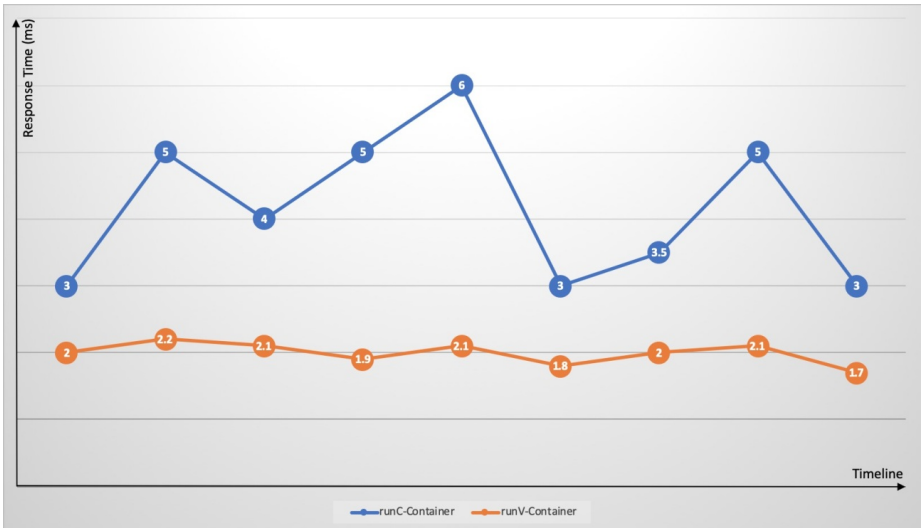
    The preceding measures can enhance the security of containers in runC and reduce attacks on the host kernel by malicious containers. However, container escapes and host kernel vulnerabilities remain unresolved.

- Sandboxed-Container prevents potential risks based on container isolation



In a Sandboxed-Container runtime environment, applications that have potential security risks are deployed on sandboxed and lightweight virtual machines. Each of the virtual machines has a dedicated guest OS kernel. If a security vulnerability is detected on a guest OS kernel, the attack is limited to one sandbox and does not affect the host kernel or other containers. The Terway network plug-in allows you to define networking policies for pods. This enables system isolation, data isolation, and network isolation for Sandboxed-Containers.

- Scenario 2: Sandboxed-Container resolves common issues of runC containers, such as fault spreading, resource contention, and performance interference.

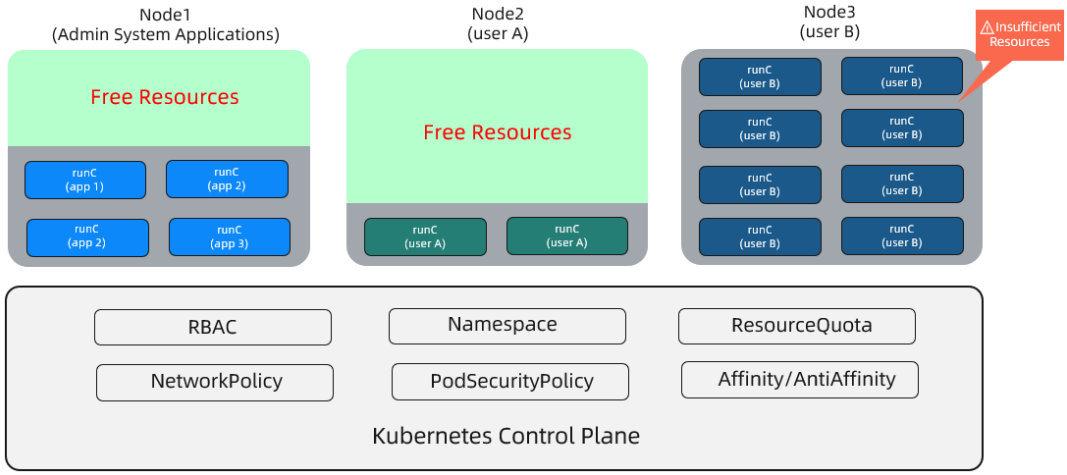


Kubernetes provides easy deployment of different containers on a single node. However, cgroups are not optimized to address resource contention. Resource-intensive applications (such as CPU-intensive, I/O-intensive applications) may compete for the same resources. This causes significant fluctuations in response time and increases the overall response time. Exceptions or faults on an application may spread to the hosting node and disrupt the running of the total cluster. For example, memory leaks and frequent core dumps of an application may overload the node, and exceptions on a container may trigger a host kernel bug that results in entire system failure. Through dedicated guest OS kernels and hypervisors, Sandboxed-Container addresses the issues that are common with runC containers. The issues include failure spreading, resource contention, and performance interference.

- Scenario 3: Sandboxed-Container supports multi-tenant services.

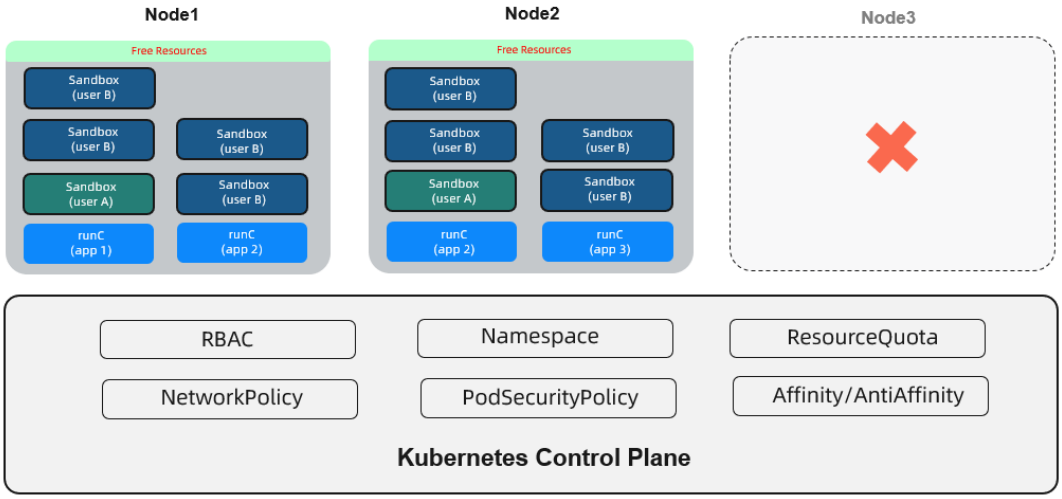
You may need isolate the applications of an enterprise that consists of multiple business lines or departments. For example, a financial department requires high security applications. However, other non-security-sensitive applications do not have high security requirements. Containers in runC fail to eliminate the potential risks brought by untrusted applications. In this scenario, you can implement the following counter measures:

- Deploy multiple independent single-tenant clusters. For example, deploy financial business and other non-security-sensitive business in different clusters.
- Deploy a multi-tenant cluster and separate applications of different business lines by namespaces. The resource of a node is exclusive to a single business line. This solution provides data isolation for coordination with the resource quotas and network policies to implement multi-tenant isolation. Compared with multiple single-tenant clusters, this solution focuses on fewer management planes and thus reduces management costs. However, this solution cannot avoid resource waste on nodes caused by low resource utilization of some tenants.



Sandboxed-Container allows you to isolate untrusted applications by using sandboxed virtual machines. This prevents the risks of container escapes. This also allows you to deploy different containerized applications on each node, which brings the following benefits:

- Resource scheduling is simplified.
- A node is no longer exclusive to a service. This improves node resource usage and reduces resource fragments and cluster resource costs.
- Sandboxed containers use lightweight virtual machines to provide almost the same performance as containers in runC.



25.5. Create a security sandbox cluster

25.5.1. Create a managed Kubernetes cluster that runs sandboxed containers

This topic describes how to create a managed Kubernetes cluster that runs sandboxed containers in the Container Service for Kubernetes (ACK) console.

Prerequisites

- ACK and Resource Access Management (RAM) are activated.
- ACK is activated in the ACK console. RAM is activated in the RAM console.

Limits

- Server Load Balancer (SLB) instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
- ACK clusters support only VPCs.
- By default, each account has specific quotas on cloud resources that can be created. You cannot create clusters if the quota is reached. Make sure that you have sufficient quotas before you create a cluster. To request a quota increase, submit a ticket.
  - By default, you can create up to five clusters across all regions with each account. Each cluster can contain up to 40 nodes. To increase the quota of clusters or nodes, submit a ticket.

    > **Notice** By default, you can add up to 48 route entries to the VPC where an ACK cluster is deployed. This means that you can configure up to 48 route entries for ACK clusters deployed in a VPC. To increase the quota of route entries for a VPC, submit a ticket.

  - By default, you can create up to 100 security groups with each account.
  - By default, you can create up to 60 pay-as-you-go SLB instances with each account.
  - By default, you can create up to 20 elastic IP addresses (EIPs) with each account.
- To create an ACK cluster that runs sandboxed containers, you must set the parameters as described in the following table. Otherwise, the cluster cannot run sandboxed containers.

Parameter	Description
Zone	Only Elastic Compute Service (ECS) Bare Metal instances support sandboxed containers. Make sure that you can purchase ECS Bare Metal instances in the selected zone.
Kubernetes Version	Select 1.14.6-aliyun.1 or later.
Container Runtime	Select Sandboxed-Container.
Worker Instance	Add worker nodes by creating new ECS instances.
Instance Type	Select ECS Bare Metal Instance.
Mount Data Disk	Mount a data disk of at least 200 GiB. We recommend that you mount a data disk of at least 1 TB.

Parameter	Description
Operating System	By default, the AliyunLinux operating system is used. You cannot not change the operating system.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Cluster Template**.
4. In the **Select Cluster Template** dialog box, find **Standard Managed Cluster** in the **Managed Clusters** section and click **Create**.
5. On the **Managed Kubernetes** tab, configure the cluster.
   i. Configure basic settings of the cluster.

Parameter	Description
Cluster Name	Enter a name for the ACK cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
Cluster Specification	Select a cluster type. You can select **Standard edition** or **Professional**.
Region	Select a region to deploy the cluster.
All Resources	Move the pointer over **All Resources** at the top of the page and select the resource group that you want to use. After you select a resource group, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource group. When you create a cluster, only the VPCs and vSwitches that belong to the selected resource group are displayed in the console. 
Kubernetes Version	Select a Kubernetes version.
Container Runtime	Select **Sandboxed-Container**.
VPC	Select a VPC to deploy the cluster. Standard VPCs and shared VPCs are supported. ■ Shared VPC: The owner of a VPC (resource owner) can share the vSwitches in the VPC with other accounts in the same organization. ■ Standard VPC: The owner of a VPC (resource owner) cannot share the vSwitches in the VPC with other accounts. **Note** ACK clusters support only VPCs. You can select a VPC from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.

Parameter	Description
VSwitch	Select vSwitches. You can select up to three vSwitches that are deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.
Network Plug-in	Select a network plug-in. Flannel and Terway are supported. For more information, see Terway and Flannel. ■ Flannel: a simple and stable Container Network Interface (CNI) plug-in that is developed by open source Kubernetes. Flannel provides a few simple features. However, it does not support standard Kubernetes network policies. ■ Terway: a network plug-in that is developed by ACK. Terway allows you to assign elastic network interfaces (ENIs) of Alibaba Cloud to containers. It also allows you to customize Kubernetes network policies to regulate how containers communicate with each other and implement bandwidth throttling on individual containers. **Note** ■ The number of pods that can be deployed on a node depends on the number of ENIs that are attached to the node and the maximum number of secondary IP addresses that are provided by these ENIs. ■ If you select a shared VPC for an ACK cluster, you must select Terway as the network plug-in. ■ If you select **Terway**, an ENI is shared among multiple pods. A secondary IP address of the ENI is assigned to each pod.

Parameter	Description
IP Addresses per Node	If you select **Flannel** as the network plug-in, you must set **IP Addresses per Node**. Note ■ **IP Addresses per Node** specifies the maximum number of IP addresses that can be assigned to each node. We recommend that you use the default value. ■ After you select the VPC and specify the number of IP addresses per node, recommended values are automatically generated for Pod CIDR block and Service CIDR block. The system also provides the maximum number of nodes that can be deployed in the cluster and the maximum number of pods that can be deployed on each node. You can modify the values based on your business requirements.
Pod CIDR Block	If you select **Flannel** as the network plug-in, you must set **Pod CIDR Block**. The CIDR block specified by **Pod CIDR Block** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Pod VSwitch	If you select Terway as the network plug-in, you must allocate vSwitches to pods. For each vSwitch that allocates IP addresses to worker nodes, you must select a vSwitch in the same zone to allocate IP addresses to pods.
Service CIDR	Set **Service CIDR**. The CIDR block specified by **Service CIDR** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Configure SNAT	By default, an ACK cluster cannot access the Internet. If the VPC that you select for the ACK cluster cannot access the Internet, you can select **Configure SNAT for VPC**. This way, ACK will create a NAT gateway and configure Source Network Address Translation (SNAT) rules to enable Internet access for the VPC.

Parameter	Description
Access to API Server	By default, an internal-facing Server Load Balancer (SLB) instance is created for the cluster API server. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. **Notice** If you delete the SLB instance, you cannot access the cluster API server. Select or clear **Expose API Server with EIP**. The ACK API server provides multiple HTTP-based RESTful APIs, which can be used to create, delete, modify, query, and monitor resources, such as pods and Services. ■ If you select this check box, an elastic IP address (EIP) is created and associated with an Internet-facing SLB instance. Port 6443 used by the API server is opened on master nodes. You can connect to and manage the ACK cluster by using kubeconfig over the Internet. ■ If you clear this check box, no EIP is created. You can connect to and manage the ACK cluster by using kubeconfig only within the VPC.
RDS Whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the cluster to the RDS whitelist. **Note** To enable an RDS instance to access the cluster, you must deploy the RDS instance in the VPC where the cluster is deployed.
Security Group	You can select **Create Basic Security Group**, **Create Advanced Security Group**, or **Select Existing Security Group**. For more information, see Overview. **Note** To select **Select Existing Security Group**, Submit a ticket to apply to be added to a whitelist.
Deletion Protection	

ii. Configure advanced settings of the cluster.

Parameter	Description
Time Zone	Select a time zone for the ACK cluster. By default, the time zone configured for your browser is selected.
Kube-proxy Mode	iptables and IPVS are supported. ■ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the ACK cluster. This mode is suitable for ACK clusters that manage a small number of Services. ■ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for ACK clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.

Parameter	Description
Labels	Add labels to the cluster. Enter a key and a value, and click **Add**. **Note** ■ Key is required. *Value* is optional. ■ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ■ *Values* are not case-sensitive. A value can be empty and can contain up to 128 characters in length. It cannot be http:// or https://. ■ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the others that use the same key. ■ If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Cluster Domain	Set the domain name of the cluster. **Note** The default domain name is **cluster.local**. You can enter a custom domain name. A domain name consists of two parts. Each part must be 1 to 63 characters in length and can contain only letters and digits. You cannot leave these parts empty.
Custom Certificate SANs	You can specify custom subject alternative names (SANs) for the API server certificate. Separate multiple IP addresses or domain names with commas (,). For more information, see Customize the SAN of the API server certificate for a managed Kubernetes cluster.
Service Account Token Volume Projection	**Service account token volume projection** reduces security risks when pods use service accounts to access the API server. This feature enables kubelet to request and store the token on behalf of the pod. This feature also allows you to configure token properties, such as the audience and validity duration. For more information, see Enable service account token volume projection.

6. Click **Next: Worker Configurations** to configure worker nodes.

Note To create an ACK cluster that runs sandboxed containers, you must select ECS Bare Metal instances as worker nodes.

i. Configure basic settings of worker nodes.

Parameter	Description
Worker Instance	By default, **Create Instance** is selected. You cannot select **Add Existing Instance**.

Parameter	Description
Billing Method	The **pay-as-you-go** and **subscription** billing methods are supported. If you select the **subscription** billing method, you must set the following parameters: ■ **Duration**: You can select 1, 2, 3, or 6 months. If you require a longer duration, you can select 1 to 5 years. ■ **Auto Renewal**: Specify whether to enable auto-renewal.
Instance Type	Select **ECS Bare Metal Instance**. Only ECS Bare Metal instances are supported.
Selected Types	The selected instance type is displayed. You can select only ECS Bare Metal Instance as the instance type.
Quantity	Specify the number of worker nodes (ECS instances) to be created.
System Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. **Note** ■ You can select **Enable Backup** to back up disk data. ■ If you select **enhanced SSD** as the system disk type, you can set a custom **performance level** for the system disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.
Mount Data Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. You can enable **disk encryption** and **disk backup** when you mount a data disk. **Notice** The data disk is used to store the root file system of all containers on the node. Therefore, you must mount a disk of at least 200 GiB. We recommend that you mount a disk of at least 1 TB.
Operating System	By default, the AliyunLinux operating system is used. You cannot not change the operating system.

Parameter	Description
Logon Type	■ Key pair logon ■ **Key Pair**: Select an SSH key pair from the drop-down list. ■ **create a key pair**: Create an SSH key pair if none is available. For more information about how to create an SSH key pair, see Create an SSH key pair. After the key pair is created, set it as the credential that is used to log on to the cluster. ■ Password logon ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.

ii. Configure advanced settings of the worker nodes.

Parameter	Description
Node Protection	Specify whether to enable node protection. **Note** By default, this check box is selected. Node protection prevents nodes from being accidentally deleted in the console or by calling the API. This prevents user errors.
User Data	For more information, see Overview of ECS instance user data.
Custom Node Name	Specify whether to use a **custom node name**. A node name consists of a prefix, an IP substring, and a suffix. ■ Both the prefix and suffix can contain one or more parts that are separated by periods (.).These parts can contain lowercase letters, digits, and hyphens (-), and must start and end with a lowercase letter or digit. ■ The IP substring length specifies the number of digits to be truncated from the end of the returned node IP address. Valid values: 5 to 12. For example, if the node IP address is 192.1xx.x.xx, the prefix is aliyun.com, the IP substring length is 5, and the suffix is test, the node name will be aliyun.com00055test.
CPU Policy	Set the CPU policy. ■ none: This policy indicates that the default CPU affinity is used. This is the default policy. ■ static: This policy allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Taints	Add taints to the worker nodes in the ACK cluster.

7. Click **Next: Component Configurations** to configure components.

Parameter	Description

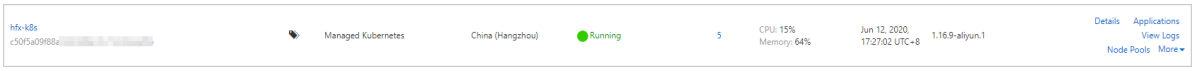
Parameter	Description
Ingress	Specify whether to install Ingress controllers. By default, **Install Ingress Controllers** is selected. For more information, see Ingress高级用法. **Note** If you want to select **Create Ingress Dashboard**, you must first enable Log Service.
Volume Plug-in	Only CSI is supported by ACK clusters that run sandboxed containers. An ACK cluster can be automatically bound to cloud disks of Alibaba Cloud, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets that are mounted to pods in the cluster. For more information, see Storage management-CSI.
Monitoring Agents	Specify whether to install the CloudMonitor agent. By default, **Install CloudMonitor Agent on ECS Instance** and **Enable Prometheus Monitoring** are selected. After the CloudMonitor agent is installed on ECS nodes, you can view monitoring data about the nodes in the CloudMonitor console.
Alerts	Select **Use Default Alert Template** to enable the alerting feature and use the default alert rules. For more information, see Alert management.
Log Service	Specify whether to enable Log Service. You can select an existing Log Service project or create one. By default, **Enable Log Service** is selected. When you create an application, you can enable Log Service through a few steps. For more information, see Collect log files from containers by using Log Service. By default, **Install node-problem-detector and Create Event Center** is selected. You can also specify whether to **create Ingress dashboards** in the Log Service console.
Log Collection for Control Plane Components	If you select **Enable**, log of the control plane components is collected to the specified Log Service project that belongs to the current account. For more information, see Collect the logs of control plane components in a managed Kubernetes cluster.
Workflow Engine	Specify whether to enable Alibaba Cloud Genomics Service (AGS). **Note** To use this feature, submit a ticket to apply to be added to a whitelist. ○ If you select this check box, the system automatically installs the AGS workflow plug-in when the system creates the cluster. ○ If you clear this check box, you must manually install the AGS workflow plug-in. For more information, see Introduction to AGS CLI.

8. Click **Next: Confirm Order**.
9. Select **Terms of Service** and click **Create Cluster**.

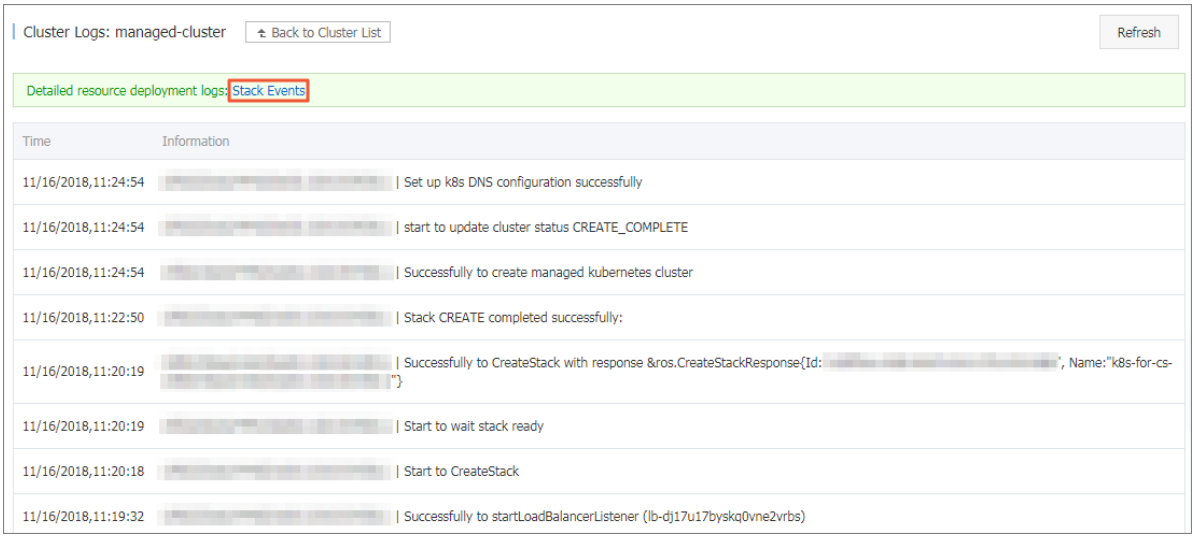
Note It requires approximately 10 minutes for the system to create a managed Kubernetes cluster that contains multiple nodes.

Result

- After the cluster is created, you can find the created cluster on the **Clusters** page in the ACK console.



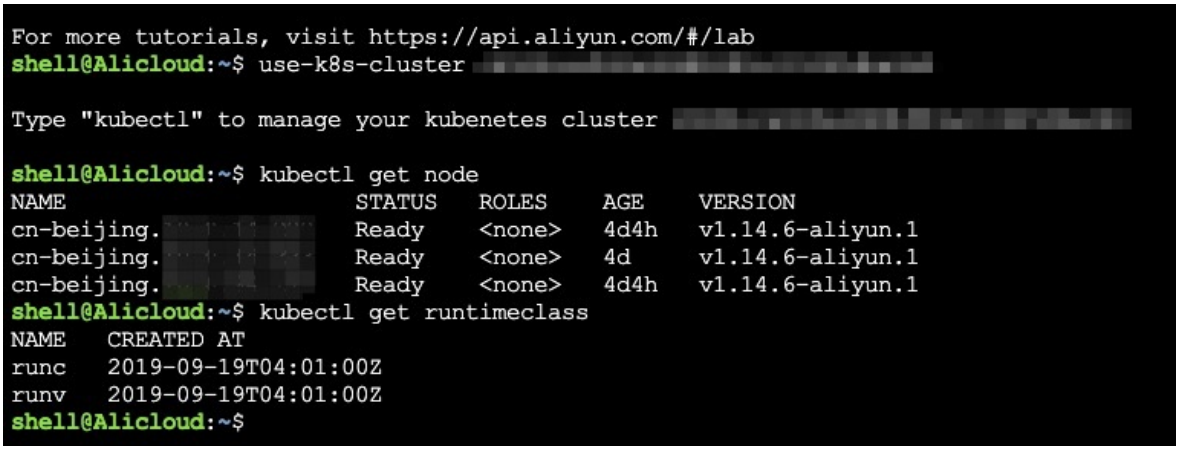
- Click **View Logs** in the **Actions** column. On the **Log Information** page, you can view log data of the cluster. To view detailed log data, click **Stack events**.



- Click **Details** in the **Actions** column. On the details page of the cluster, click the **Basic Information** tab to view basic information about the cluster and click the **Connection Information** tab to view information about how to connect to the cluster.
  The following information is displayed.
  - **API Server Public Endpoint**: the IP address and port that the API server uses to provide services over the Internet. It allows you to manage the cluster by using kubectl or other tools on your terminals.
    Bind EIP and **Unbind EIP**: These options are available to only managed Kubernetes clusters.
    - Bind EIP: You can select an existing elastic IP address (EIP) or create one.
      The API server restarts after you bind an EIP to the API server. We recommend that you do not perform operations on the cluster during the restart process.
    - Unbind EIP: You cannot access the API server over the Internet after you unbind the EIP.
      The API server restarts after you unbind the EIP from the API server. We recommend that you do not perform operations on the cluster during the restart process.
  - **API Server Internal Endpoint**: the IP address and port that the API server uses to provide services within the cluster. The IP address belongs to the Server Load Balancer (SLB) instance that is bound to the cluster.
  - **Testing Domain**: the domain name that is used to test Services. The suffix of the domain name is `<cluster_id>.<region_id>.alicontainer.com`.

    > **Note** To rebind the domain name, click **Rebind Domain Name**.

- You can Connect to Kubernetes clusters by using kubectl and run the `kubectl get node` command to query information about the nodes in the cluster.



25.5.2. Create a dedicated Kubernetes cluster that supports sandboxed containers

This topic describes how to create a dedicated Kubernetes cluster that supports sandboxed containers in the Container Service for Kubernetes (ACK) console.

Prerequisites

Related cloud services are activated. For more information, see Quick start for first-time users.

Limits

- Server Load Balancer (SLB) instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
- ACK clusters support only VPCs.
- By default, each account is subject to specific quotas on cloud resources that can be created. You cannot create clusters if the quota limit is exceeded. Make sure that you have sufficient resource quotas before you create a cluster. To increase the quota of cloud resources for your account, submit a ticket.
  - By default, you can create up to five clusters across all regions with each account. Each cluster can contain up to 40 nodes. To increase the quota of clusters or nodes, submit a ticket.

    > **Notice** By default, you can add up to 48 route entries to the virtual private cloud (VPC) where an ACK cluster is deployed. This means that you can configure up to 48 route entries for ACK clusters deployed in a VPC. To increase the quota of route entries for a VPC, submit a ticket.

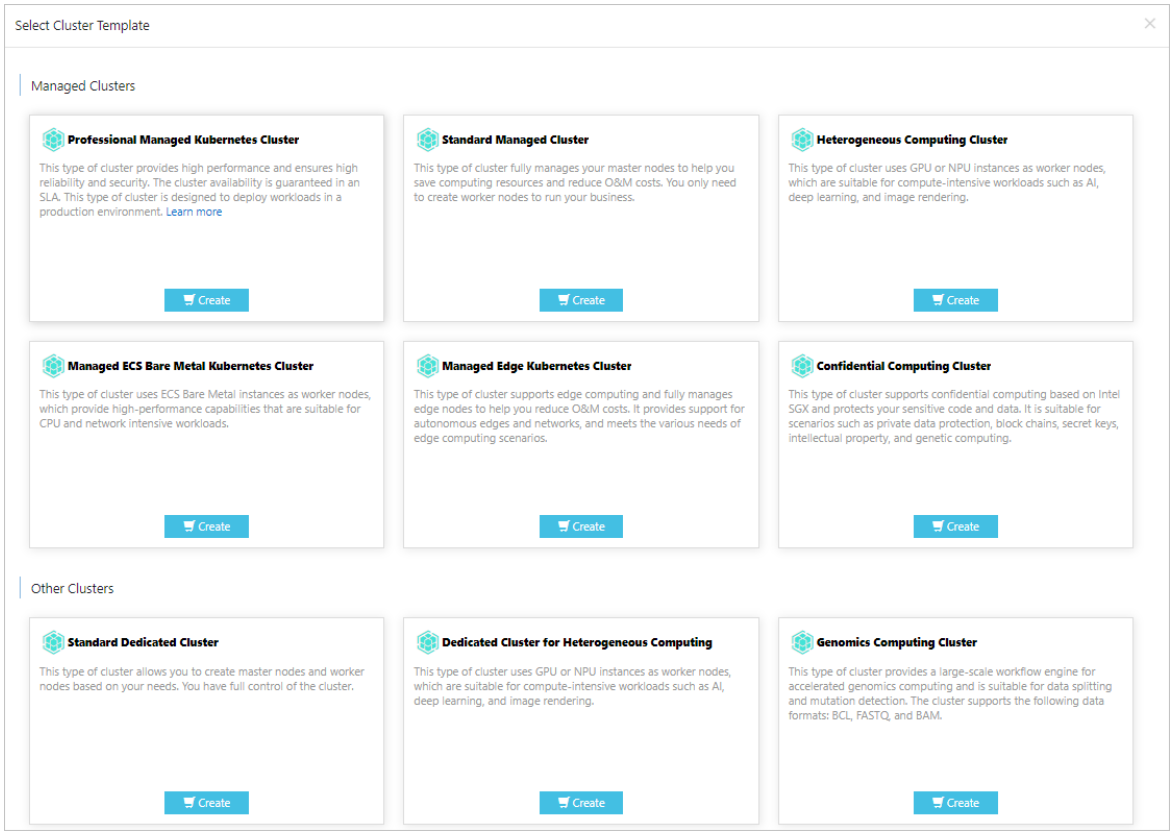
  - By default, you can create up to 100 security groups with each account.
  - By default, you can create up to 60 pay-as-you-go SLB instances with each account.
  - By default, you can create up to 20 elastic IP addresses (EIPs) with each account.
- To create an ACK cluster that runs sandboxed containers, you must set the parameters as described in the following table. Otherwise, the cluster cannot run sandboxed containers.

Parameter	Description
Zone	Only Elastic Compute Service (ECS) Bare Metal instances support sandboxed containers. Make sure that you can purchase ECS Bare Metal instances in the selected zone.

Parameter	Description
Kubernetes Version	Select 1.14.6-aliyun.1 or later.
Container Runtime	Select Sandboxed-Container.
Worker Instance	Add worker nodes by creating new ECS instances.
Instance Type	Select ECS Bare Metal Instance.
Mount Data Disk	Mount a data disk of at least 200 GiB. We recommend that you mount a data disk of at least 1 TB.
Operating System	By default, the AliyunLinux operating system is used. You cannot not change the operating system.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Cluster Template**.
4. In the **Select Cluster Template** dialog box, find **Standard Dedicated Cluster** in the **Other Clusters** section and click **Create**.



5. On the **Dedicated Kubernetes** tab, configure the cluster.
   i. Configure basic settings of the cluster.

Parameter	Description
Cluster Name	Enter a name for the ACK cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
Region	Select a region to deploy the cluster.
Resource Group	Move the pointer over **All Resources** at the top of the page and select the resource group that you want to use. After you select a resource group, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource group. When you create a cluster, only the VPCs and vSwitches that belong to the selected resource group are displayed in the console. 
Kubernetes Version	Select a Kubernetes version.
Container Runtime	Select **Sandboxed-Container**.
VPC	Select a VPC to deploy the cluster. Standard VPCs and shared VPCs are supported. ■ Shared VPC: The owner of a VPC (resource owner) can share the vSwitches in the VPC with other accounts in the same organization. ■ Standard VPC: The owner of a VPC (resource owner) cannot share the vSwitches in the VPC with other accounts. **Note** ACK clusters support only VPCs. You can select a VPC from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.
VSwitch	Select vSwitches. You can select up to three vSwitches that are deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.

Parameter	Description
Network Plug-in	Select a network plug-in. Flannel and Terway are supported. For more information, see Terway and Flannel. ■ Flannel: a simple and stable Container Network Interface (CNI) plug-in that is developed by open source Kubernetes. Flannel provides a few simple features. However, it does not support standard Kubernetes network policies. ■ Terway: a network plug-in that is developed by ACK. Terway allows you to assign elastic network interfaces (ENIs) of Alibaba Cloud to containers. It also allows you to customize Kubernetes network policies to regulate how containers communicate with each other and implement bandwidth throttling on individual containers. Note ■ The number of pods that can be deployed on a node depends on the number of ENIs that are attached to the node and the maximum number of secondary IP addresses that are provided by these ENIs. ■ If you select a shared VPC for an ACK cluster, you must select Terway as the network plug-in. ■ If you select **Terway**, an ENI is shared among multiple pods. A secondary IP address of the ENI is assigned to each pod.
IP Addresses per Node	If you select **Flannel** as the network plug-in, you must set **IP Addresses per Node**. Note ■ **IP Addresses per Node** specifies the maximum number of IP addresses that can be assigned to each node. We recommend that you use the default value. ■ After you select the VPC and specify the number of IP addresses per node, recommended values are automatically generated for Pod CIDR block and Service CIDR block. The system also provides the maximum number of nodes that can be deployed in the cluster and the maximum number of pods that can be deployed on each node. You can modify the values based on your business requirements.

Parameter	Description
Pod CIDR Block	If you select **Flannel** as the network plug-in, you must set **Pod CIDR Block**. The CIDR block specified by **Pod CIDR Block** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Pod VSwitch	If you select Terway as the network plug-in, you must allocate vSwitches to pods. For each vSwitch that allocates IP addresses to worker nodes, you must select a vSwitch in the same zone to allocate IP addresses to pods. The elastic network interfaces (ENI) used by a pod must be deployed in the zone where the node that hosts the pod is deployed. Therefore, you must select another vSwitch in the zone where the node vSwitch is deployed. This vSwitch assigns IP addresses to pods. To ensure sufficient IP addresses for all pods, we recommend that you set the mask length of the CIDR block to a value no greater than 19 for the vSwitch.
Service CIDR	Set **Service CIDR**. The CIDR block specified by **Service CIDR** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Configure SNAT	By default, an ACK cluster cannot access the Internet. If the VPC that you select for the ACK cluster cannot access the Internet, you can select **Configure SNAT for VPC**. This way, ACK will create a NAT gateway and configure Source Network Address Translation (SNAT) rules to enable Internet access for the VPC.
Access to API Server	By default, an internal-facing Server Load Balancer (SLB) instance is created for the cluster API server. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. **Notice** If you delete the SLB instance, you cannot access the cluster API server. Select or clear **Expose API Server with EIP**. The ACK API server provides multiple HTTP-based RESTful APIs, which can be used to create, delete, modify, query, and monitor resources, such as pods and Services. ■ If you select this check box, an elastic IP address (EIP) is created and associated with an Internet-facing SLB instance. Port 6443 used by the API server is opened on master nodes. You can connect to and manage the ACK cluster by using kubeconfig over the Internet. ■ If you clear this check box, no EIP is created. You can connect to and manage the ACK cluster by using kubeconfig only within the VPC.

Parameter	Description
SSH Logon	To enable SSH logon, you must first select **Expose API Server with EIP**. ■ If you select Use SSH to Access the Cluster from the Internet, you can access the cluster by using SSH. ■ If you clear this check box, you cannot access the cluster by using SSH or kubectl. If you want to access an Elastic Compute Service (ECS) instance in the cluster by using SSH, you must manually associate an elastic IP address (EIP) with the instance and configure security group rules to open SSH port 22. For more information, see Use SSH to connect to an ACK cluster.
RDS Whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the cluster to the RDS whitelist. **Note** To enable an RDS instance to access the cluster, you must deploy the RDS instance in the VPC where the cluster is deployed.
Security Group	You can select **Create Basic Security Group**, **Create Advanced Security Group**, or **Select Existing Security Group**. For more information, see Overview. **Note** To select **Select Existing Security Group**, Submit a ticket to apply to be added to a whitelist.

ii. Configure advanced settings of the cluster.

Parameter	Description
Time Zone	Select a time zone for the ACK cluster. By default, the time zone configured for your browser is selected.
Kube-proxy Mode	iptables and IPVS are supported. ■ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the ACK cluster. This mode is suitable for ACK clusters that manage a small number of Services. ■ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for ACK clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.

Parameter	Description
Labels	Add labels to the cluster. Enter a key and a value, and click **Add**. **Note** ■ Key is required. *Value* is optional. ■ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ■ *Values* are not case-sensitive. A value can be empty and can contain up to 128 characters in length. It cannot be http:// or https://. ■ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the others that use the same key. ■ If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Cluster Domain	Set the domain name of the cluster. **Note** The default domain name is **cluster.local**. You can enter a custom domain name. A domain name consists of two parts. Each part must be 1 to 63 characters in length and can contain only letters and digits. You cannot leave these parts empty.
Custom Certificate SANs	You can enter custom subject alternative names (SANs) for the API server certificate of the cluster to accept requests from specified IP addresses or domain names.
Service Account Token Volume Projection	**Service account token volume projection** reduces security risks when pods use service accounts to access the API server. This feature enables kubelet to request and store the token on behalf of the pod. This feature also allows you to configure token properties, such as the audience and validity duration. For more information, see Enable service account token volume projection.
Cluster CA	If you select this check box, upload a Certificate Authority (CA) certificate for the ACK cluster to ensure secure data transmission between the server and client.
Deletion Protection	

6. Click **Next: Master Configurations** to configure master nodes.

Parameter	Description
Billing Method	The **pay-as-you-go** and **subscription** billing methods are supported. If you select **pay-as-you-go**, you must set the following parameters: ○ **Duration**: You can select 1, 2, 3, or 6 months. If you require a longer duration, you can select 1 to 5 years. ○ **Auto Renewal**: Specify whether to enable auto-renewal.
Master Node Quantity	Specify the number of master nodes. You can create three or five master nodes.
Instance Type	Select an instance type for the master nodes. For more information, see Instance families.
System Disk	By default, system disks are mounted to master nodes. **Standard SSDs** and **ultra disks** are supported. **Note** You can select **Enable Backup** to back up disk data.

7. Click **Next: Worker Configurations** to configure worker nodes.
   i. Configure basic settings of worker nodes.

Parameter	Description
Worker Instance	By default, **Create Instance** is selected. You cannot select **Add Existing Instance**.
Instance Type	Select **ECS Bare Metal Instance**. Only ECS bare metal instances are supported.
Selected Types	The selected instance type is displayed. You can select only one instance type of ECS Bare Metal Instance.
Quantity	Specify the number of worker nodes (ECS instances) to be created.
System Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. **Note** ■ You can select **Enable Backup** to back up disk data. ■ If you select **enhanced SSD** as the system disk type, you can set a custom **performance level** for the system disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.

Parameter	Description
Mount Data Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. You can enable **disk encryption** and **disk backup** when you mount a data disk. **Notice** The data disk is used to store the root file systems of all containers on the node. Therefore, you must mount a data disk of at least 200 GiB. We recommend that you mount a data disk of at least 1 TB.
Operating System	By default, the AliyunLinux operating system is used. You cannot change the operating system.
Logon Type	■ Key pair logon ■ **Key Pair**: Select an SSH key pair from the drop-down list. ■ **create a key pair**: Create an SSH key pair if none is available. For more information about how to create an SSH key pair, see Create an SSH key pair. After the key pair is created, set it as the credential that is used to log on to the cluster. ■ Password logon ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.

ii. Configure advanced settings of worker nodes

Parameter	Description
Node Protection	Specify whether to enable node protection. **Note** By default, this check box is selected. Node protection prevents nodes from being accidentally deleted in the console or by calling the API. This prevents user errors.
User Data	For more information, see Overview of ECS instance user data.
Custom Node Name	Specify whether to use a **custom node name**. A node name consists of a prefix, an IP substring, and a suffix. ■ Both the prefix and suffix can contain one or more parts that are separated by periods (.).These parts can contain lowercase letters, digits, and hyphens (-), and must start and end with a lowercase letter or digit. ■ The IP substring length specifies the number of digits to be truncated from the end of the returned node IP address. Valid values: 5 to 12. For example, if the node IP address is 192.1xx.x.xx, the prefix is aliyun.com, the IP substring length is 5, and the suffix is test, the node name will be aliyun.com00055test.
Node Port Range	Set the node port range.
CPU Policy	Set the CPU policy. ■ none: This policy indicates that the default CPU affinity is used. This is the default policy. ■ static: This policy allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Taints	Add taints to the worker nodes in the ACK cluster.

8. Click **Next:Component Configurations** to configure components.

Parameter	Description
Ingress	Specify whether to install Ingress controllers. By default, **Install Ingress Controllers** is selected. For more information, see Ingress高级用法.
Volume Plug-in	Select **CSI**. Only the Container Storage Interface (CSI) plug-in is supported by sandboxed containers in ACK clusters. ACK clusters can be automatically bound to Alibaba Cloud disks, Apsara File Storage NAS (NAS) file systems, and Object Storage Service (OSS) buckets that are mounted to pods. For more information, see Storage management-CSI.
Monitoring Agents	Specify whether to install the CloudMonitor agent. By default, **Install CloudMonitor Agent on ECS Instance** and **Enable Prometheus Monitoring** are selected. After the CloudMonitor agent is installed on ECS nodes, you can view monitoring data about the nodes in the CloudMonitor console.

Parameter	Description
Log Service	Specify whether to enable Log Service. You can select an existing Log Service project or create one. By default, **Enable Log Service** is selected. When you create an application, you can enable Log Service through a few steps. For more information, see Collect log files from containers by using Log Service. By default, **Install node-problem-detector and Create Event Center** is selected. You can also specify whether to **create Ingress dashboards** in the Log Service console.
Workflow Engine	Specify whether to enable Alibaba Cloud Genomics Service (AGS). **Note** To use this feature, submit a ticket to apply to be added to a whitelist. ○ If you select this check box, the system automatically installs the AGS workflow plug-in when the system creates the cluster. ○ If you clear this check box, you must manually install the AGS workflow plug-in. For more information, see Introduction to AGS CLI.

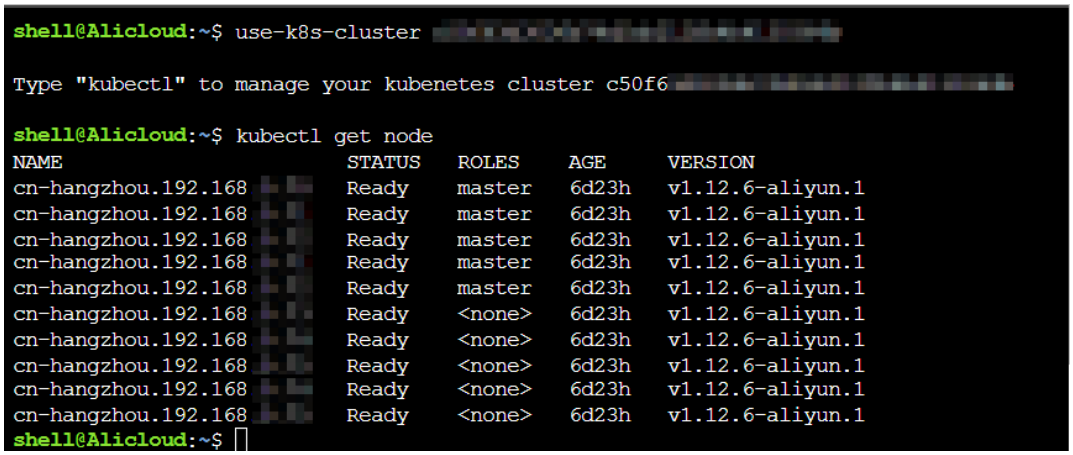
9. Click **Next: Confirm Order**.
10. Select **Terms of Service** and click **Create Cluster**.

Note It requires approximately 10 minutes for the system to create a managed Kubernetes cluster that contains multiple nodes.

Result

- After the cluster is created, you can view the newly created cluster on the **Clusters** page in the ACK console.
- Click **View Logs** in the Actions column. On the page that appears, you can view the cluster log. To view detailed log information, click **Stack events**.
- On the Clusters page, find the newly created cluster and click **Details** in the **Actions** column. On the details page of the cluster, click the **Basic Information** tab to view basic information about the cluster and click the **Connection Information** tab to view information about how to connect to the cluster. The following information is displayed:
  - **API Server Public Endpoint**: the IP address and port that the API server uses to provide services over the Internet. It allows you to manage the cluster by using kubectl or other tools on the client.
  - **API Server Internal Endpoint**: the IP address and port that the API server uses to provide services within the cluster. The endpoint belongs to the SLB instance that is bound to the cluster. Three master nodes work as the backend servers of the SLB instance.
  - **Testing Domain**: the domain name that is used to test Services. The suffix of the domain name is `<cluster_id>.<region_id>.alicontainer.com`.

    Note To rebind the domain name, click **Rebind Domain Name**.
- You can Connect to Kubernetes clusters by using kubectl and run the `kubectl get node` command to view information about the nodes in the cluster.



25.6. Expand a cluster that runs sandboxed containers

This topic describes how to scale out the number of worker nodes in a Container Service for Kubernetes (ACK) cluster that runs sandboxed containers by using the ACK console.

Prerequisites

You cannot scale out the number of master nodes in a cluster that runs sandboxed containers.

When you expand a cluster that runs sandboxed containers, you must set the parameters as required in the following table. Otherwise, the added nodes cannot run sandboxed containers.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to expand and choose **More > Expand** in the **Actions** column.
4. On the **Node Pools** page, select the node pool that you want to scale out and click **Scale Out** in the **Actions** column.
5. In the dialog box that appears, set the parameters and click **Submit**.

   In this example, the number of worker nodes in the cluster is increased from three to five. The following table describes the required parameters.

Parameter	Description
Nodes to Add	The number of nodes to be added to the cluster.
Region	By default, the region where the cluster is deployed is displayed.
Container Runtime	By default, Sandboxed-Container is displayed.
VPC	Set the network for the nodes. You can select a virtual private cloud (VPC) from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.

Parameter	Description
VSwitch	Select one or more vSwitches. You can select up to three vSwitches deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.
Billing Method	ACK supports the **pay-as-you-go** and **subscription** billing methods. If you select the **subscription** billing method, ○ you must set the **duration**. You can select 1, 2, 3, or 6 months. If you require a longer duration, you can select 1 to 5 years. ○ you must specify whether to enable **Auto Renewal**.
Instance Type	Select **ECS Bare Metal Instance**.
Selected Types	The selected instance types.
System Disk	Standard SSDs, enhanced SSDs (ESSDs), and ultra disks are supported.
Mount Data Disk	Standard SSDs, ESSDs, and ultra disks are supported. **Note** You can specify whether to encrypt the mounted data disks. The disks are used to store the root file systems of containers on the nodes. Therefore, you must mount a data disk of at least 200 GiB. We recommend that you mount a data disk of at least 1 TB.
Logon Type Key Pair	○ Key pair: When you created the cluster, you selected Key Pair as the logon method. If no key pair is available, click **create a key pair** to create one in the Elastic Compute Service (ECS) console. For more information, see Create an SSH key pair. After the key pair is created, set it as the credentials to log on to the cluster. ○ Password: ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.
Public IP	If you select **Assign a Public IPv4 Address to Each Node**, public IPv4 addresses are automatically assigned to the new nodes. You can connect to the nodes through the assigned IPv4 addresses. For more information about public IP addresses, see Elastic IP Addresses.

Parameter	Description
ECS Label	Attach labels to the added ECS instances. **Note** ◦ Key is required. *Value* is optional. ◦ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ◦ *Values* are not case-sensitive. A value must be 1 to 128 characters in length, and cannot start with http:// or https://. ◦ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the one that uses the same key. ◦ You can add at most 20 labels to each resource. If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.

Parameter	Description
Node Label	Add labels to the added nodes. **Note** ◦ Key is required. *Value* is optional. ◦ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ◦ *Values* are not case-sensitive. A value must be 1 to 128 characters in length, and cannot start with http:// or https://. ◦ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the one that uses the same key. ◦ You can add at most 20 labels to each resource. If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Resource Group	Specify the resource group of the new nodes.
Taints	Add taints to the added nodes.
RDS Whitelist	Set the RDS whitelist. This operation adds the IP addresses of the nodes to the RDS whitelist.
CloudMonitor Agent	You can install the CloudMonitor agent on the nodes and view monitoring information about the nodes in the CloudMonitor console.
User Data	For more information, see Overview of ECS instance user data.

What's next

After the nodes are added, go to the details page of the cluster. In the left-side navigation pane, click **Node Pools**. Verify that the number of worker nodes is increased from three to five.

25.7. Create an application that runs in sandboxed containers

This topic describes how to use an image to create an NGINX application that runs in sandboxed containers. The NGINX application is accessible over the Internet.

Prerequisites

A cluster of Container Service for Kubernetes (ACK) that runs sandboxed containers is created. For more information, see Create a managed Kubernetes cluster that runs sandboxed containers.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** tab, click **Create from Image**.
6. On the **Basic Information** wizard page, set the basic information and parameters for the application, and then click **Next**.

   Set **Name**, **Replicas**, **Type**, **Annotations**, and **Label**. Specify whether to **synchronize timezone from node to container**. The number of replicas equals the number of replicated pods provisioned for the application.

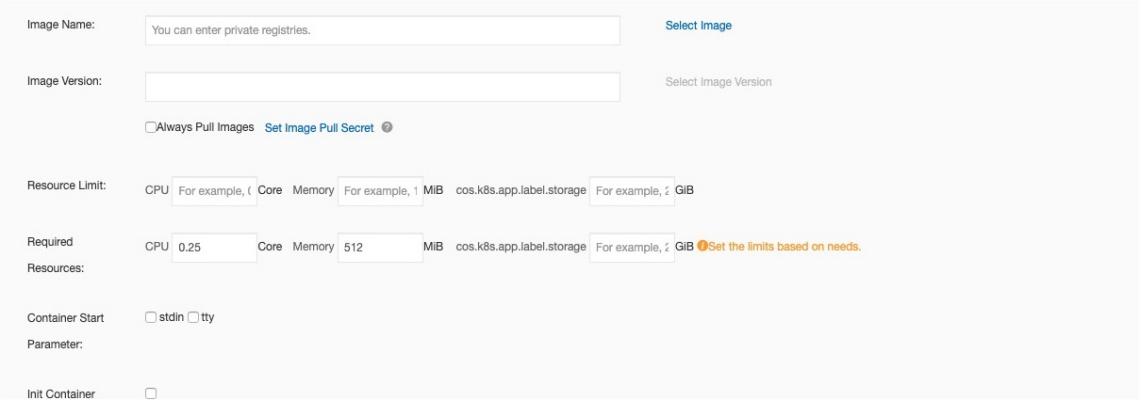
   > **Note**
   > In this example, **Deployment** is selected as the application type.

7. Configure the containers.

   > **Note** At the top of the **Container** wizard page, click **Add Container** to add more containers for the application.

   The following parameters are required to configure the containers.

   - General



Parameter	Description
Image Name	Click **Select Image**. In the dialog box that appears, select an image and click **OK**. In this example, the NGINX image is selected. You can also enter the address of a private registry. The registry address must be in the following format: `domainname/namespace/imagename:tag`.

Parameter	Description
Image Version	■ Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used. ■ You can select the following image pull policies: ■ **ifNotPresent**: If the image that you want to pull is found on your on-premises machine, the image on your on-premises machine is used. Otherwise, ACK pulls the image from the corresponding repository. ■ **Always**: ACK pulls the image from Container Registry each time the application is deployed or scaled out. ■ **Never**: ACK uses only images on your on-premises machine. **Note** If you select **Image Pull Policy**, no image pull policy is applied. ■ To pull the image without a password, click **Set Image Pull Secret** to set a Secret that is used to pull the image. For more information, see Use aliyun-acr-credential-helper to pull images without a password.
Resource Limit	You can specify an upper limit for the CPU, memory, and ephemeral storage resources that the container can consume. This prevents the container from occupying an excessive amount of resources. The CPU resource is measured in milicores (one thousandth of one core). The memory resource is measured in MiB. The ephemeral storage resource is measured in GiB.
Required Resources	The amount of CPU and memory resources that are reserved for this application. These resources are exclusive to the container. This prevents the application from becoming unavailable when other Services or processes occupy these resources.
Container Start Parameter	■ stdin: specifies that start parameters defined in the console are sent to the Linux system. ■ tty: specifies that start parameters defined in a virtual terminal are sent to the console.
Privileged Container	■ If you select Privileged Container, privileged=true is set for the container and the privilege mode is enabled. ■ If you do not select Privileged Container, privileged=false is set for the container and the privilege mode is disabled.
Init Container	If you select Init Container, an init container is created. An init container provides tools to manage pods. For more information, see Init Containers.

- (Optional)Ports
  Specify the container port.
  - Name: Enter a name for the container port.

- Container Port: the container port that you want to open. Enter a port number from 1 to 65535.
- Protocol: Select TCP or UDP.

- (Optional)Environments
  You can set environment variables in key-value pairs for pods. Environment variables are used to apply pod configurations to containers. For more information, see Pod variables.
  - Type: the type of environment variable. You can select **Custom**, **ConfigMaps**, **Secret**, **Value/ValueFrom**, or **ResourceFieldRef**. If you select ConfigMaps or Secret as the type of environment variable, all values in the selected ConfigMaps or Secret are passed to the container environment variable. In this example, Secret is selected.
    Select **Secret** from the Type drop-down list and select a Secret from the **Value/ValueFrom** drop-down list. All values in the selected Secret are passed to the environment variable.
    In this case, the YAML file that is used to deploy the application contains the settings that reference all values in the specified Secret.
  - Variable Key: Specify the key of the environment variable.
  - Value/ValueFrom: Specify the value that is referenced by the environment variable.
- (Optional)Health Check
  Health check settings include liveness and readiness probes. Liveness probes determine when to restart the container. Readiness probes indicate whether the container is ready to accept network traffic. For more information about health checks, see Configure Liveness, Readiness, and Startup Probes.

Request type	Description

Request type	Description
HTTP	Sends an HTTP GET request to the container. You can set the following parameters: ■ Protocol: HTTP or HTTPS. ■ Path: the requested path on the server. ■ Port: the container port that you want to open. Enter a port number from 1 to 65535. ■ HTTP Header: the custom headers in the HTTP request. Duplicate headers are allowed. You can set HTTP headers in key-value pairs. ■ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time period (in seconds) that the system must wait before it can send the first probe to a launched container. Default value: 3. ■ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are sent. Default value: 10. Minimum value: 1. ■ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the timeout period (in seconds) of probes. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

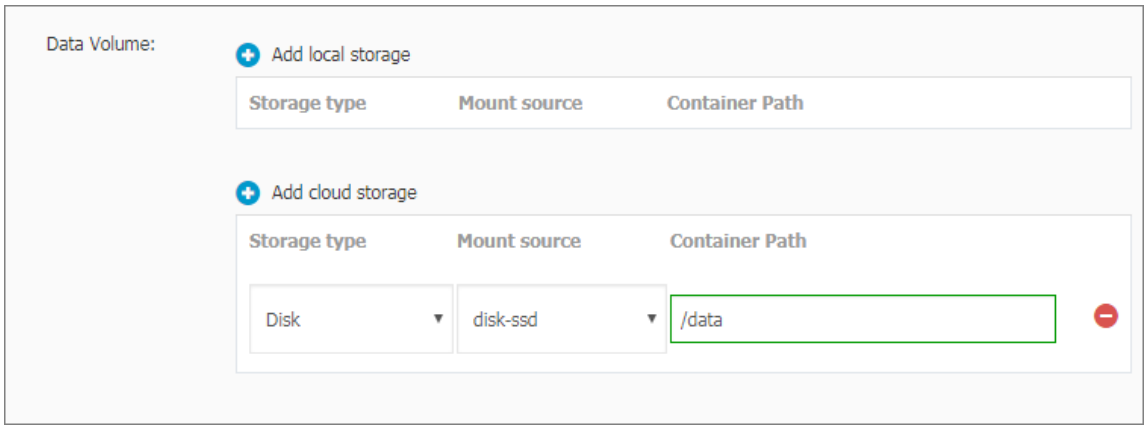
Request type	Description
TCP	Sends a TCP socket to the container. Kubelet attempts to open the socket on the specified port. If the connection can be established, the container is considered healthy. Otherwise, the container is considered unhealthy. Supported parameters include: ■ Port: the container port that you want to open. Enter a port number from 1 to 65535. ■ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time period (in seconds) that the system must wait before it can send the first probe to the launched container. Default value: 15. ■ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are sent. Default value: 10. Minimum value: 1. ■ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the timeout period (in seconds) of probes. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.
Command	Runs a probe command in the container to check the health status of the container. Supported parameters include: ■ Command: the probe command that is run to check the health status of the container. ■ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time period (in seconds) that the system must wait before it can send the first probe to the launched container. Default value: 5. ■ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are sent. Default value: 10. Minimum value: 1. ■ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the timeout period (in seconds) of probes. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

- Lifecycle
  You can set the following parameters to configure the lifecycle of the container: Start, Post Start, and Pre Stop. For more information, see Configure the lifecycle of a container.

- **Start**: Set the command and parameter that take effect before the container starts.
- **Post Start**: Set the command that takes effect after the container starts.
- **Pre Stop**: Set the command that takes effect before the container stops.

- (Optional)Volume
  You can mount local storage volumes and persistent volume claims (PVCs) to the container.
  - **Local Storage**: You can select HostPath, ConfigMap, Secret, and EmptyDir. The storage volume is mounted to a path in the container. For more information, see Volumes.
  - **PVC**: Select Cloud Storage.

  In this example, a PVC named disk-ssd is mounted to the */tmp* path of the container.



- (Optional)Log
  Configure **Log Service**. You can specify collection configurations and custom tags.

> **Notice** Make sure that the Log Service agent is installed in the cluster.

Parameter	Description
Collection Configuration	Logstore: Create a Logstore in Log Service to store collected logs.
	Log Path in Container: Specify stdout or a path to collect logs. ■ stdout: specifies that the stdout files are collected. ■ Text Logs: specifies that logs in the specified path of the container are collected. In this example, */var/log/nginx* is specified as the path. Wildcard characters can be used in this path.
Custom Tag	You can also add tags. Tags are added to the logs of the container when the logs are collected. Log data with tags is easier to aggregate and filter.

8. Set the preceding parameters based on your business requirements and click **Next**.
9. (Optional)Configure advanced settings.
   - Access Control

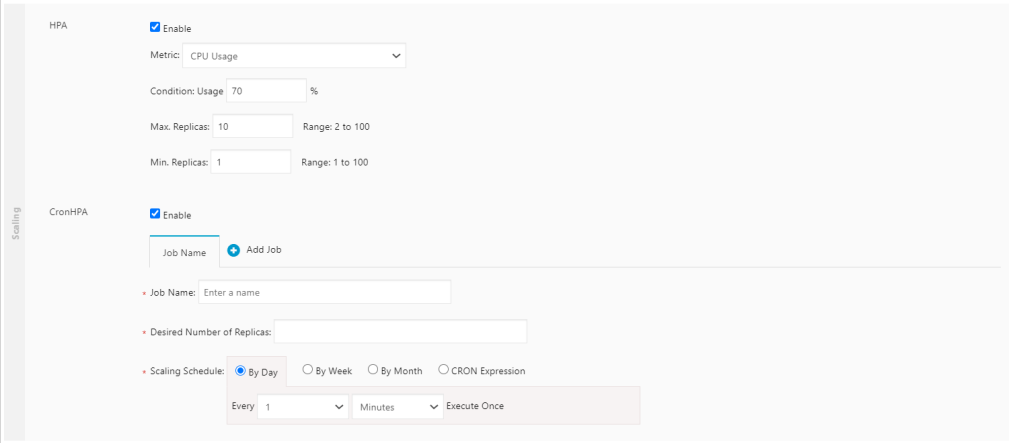
> **Note**
> You can configure the following access control settings based on your business requirements:
>
> - Internal applications: For applications that run inside the cluster, you can create a Service of the ClusterIP or NodePort type to enable internal communication.
> - External applications: For applications that are open to the Internet, you can configure access control by using one of the following methods:
>   - Create a LoadBalancer Service that uses a Server Load Balancer (SLB) instance and use the Service to expose your application to the Internet.
>   - Create an Ingress and use the Ingress to expose your application to the Internet. For more information, see Ingress.

Specify how backend pods are exposed. In this example, a ClusterIP Service and an Ingress are created to expose the NGINX application to the Internet.

Parameter	Description
Services	Click **Create** on the right side of **Services**. In the Create Service dialog box, set the parameters. For more information about the parameters that are required to create a Service, see Manage Services. **Cluster IP** is selected in this example.
Ingresses	Click **Create** on the right side of **Ingresses**. In the Create dialog box, set the parameters. For more information about the parameters that are required to create an Ingress, see Ingress configurations. **Note** When you deploy an application from an image, you can create an Ingress for only one Service. In this example, the name of a virtual host is specified as the test domain name. You must add a mapping rule for this domain name to the hosts file, as shown in the following code block. In practical scenarios, use a domain name that has obtained an Internet Content Provider (ICP) number. `101.37.224.146 foo.bar.com #The IP address of the Ingress.`

You can find the created Service and Ingress in the **Access Control** section. Click **Update** or **Delete** to modify the settings.

- Scaling
  In the **Scaling** section, specify whether to enable **HPA** and **CronHPA** based on your business requirements. Horizontal Pod Autoscaler (HPA) enables the application to run at different load levels.



- HPA can automatically scale the number of pods in a Container Service for Kubernetes (ACK) cluster based on the CPU and memory usage.

  > **Note** To enable HPA, you must configure resources that can be scaled for the container. Otherwise, HPA does not take effect.

Parameter	Description
Metric	Select CPU Usage or Memory Usage. The selected resource type must be the same as that specified in the Required Resources field.
Condition	Specify the resource usage threshold. HPA triggers scaling activities when the threshold is exceeded.
Max. Replicas	Specify the maximum number of replicated pods to which the application can be scaled.
Min. Replicas	Specify the minimum number of replicated pods that must run.

- CronHPA can scale an ACK cluster at a scheduled time. For more information about CronHPA, see Create CronHPA jobs.

- Scheduling
  You can set the following parameters: Update Method, Node Affinity, Pod Affinity, Pod Anti Affinity, and Toleration. For more information, see Affinity and anti-affinity.

  > **Note** During pod scheduling, the labels of a node and a pod determine the affinities of the node and pod. You can configure node affinity and pod affinity by selecting preset labels or by manually adding labels.

Parameter	Description
Update Method	Select Rolling Update or OnDelete. For more information, see Deployments.

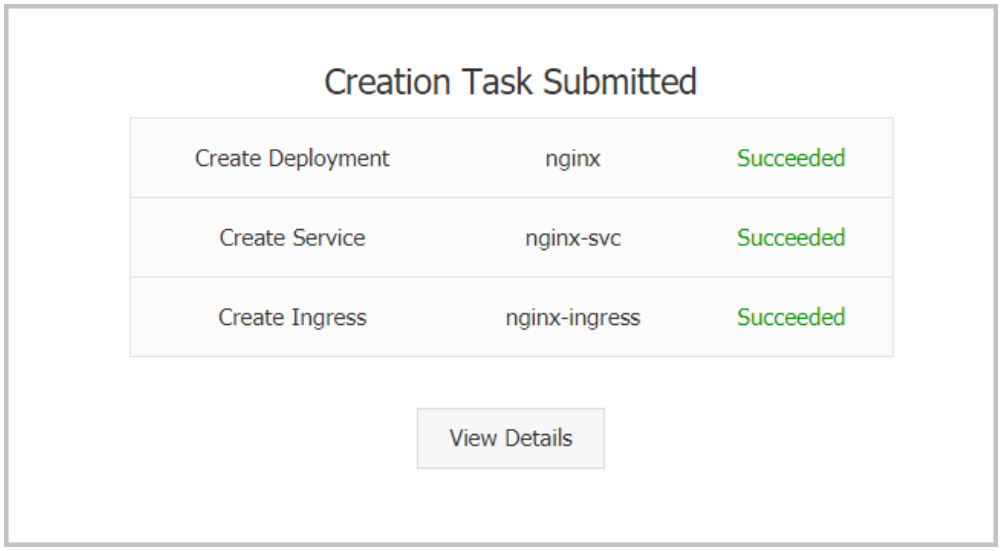
Parameter	Description
Node Affinity	Add labels to worker nodes to set **Node Affinity**. Node affinity supports required and preferred rules, and various operators, such as In, NotIn, Exists, DoesNotExist, Gt, and Lt. ■ **Required**: Specify the rules that must be matched for a pod to be scheduled to a node. Required rules correspond to the requiredDuringSchedulingIgnoredDuringExecution affinity, which is conceptually similar to `NodeSelector`. In this example, pods can be scheduled to only worker nodes with specified labels. You can create more than one required rule. However, only one required rule must be matched. ■ **Preferred**: Specify the rules that are preferred to match. Preferred rules correspond to the preferredDuringSchedulingIgnoredDuringExecution affinity. In this example, the scheduler attempts to avoid scheduling the pod to a node that matches the preferred rules. You can set node weights in preferred rules. If multiple nodes match the preferred rules, the pod is preferably scheduled to the node with the highest weight. You can create more than one preferred rule. All preferred rules must be matched before a pod can be scheduled to a preferred node.

Parameter	Description
Pod Affinity	Pod affinity specifies that a pod is scheduled to a node in the same topological domain if the node runs a pod that matches the affinity rules. For example, you can use pod affinity to deploy Services that communicate with each other to the same topological domain, such as a host. This reduces the network latency between these Services. You can enforce pod affinity by using the labels of the pods that run on a node. Pod affinity supports required and preferred rules, and the following operators: **In, NotIn, Exists, and DoesNotExist**. ■ **Required**: Specify the rules that must be matched for a pod to be scheduled to a node. Required rules correspond to the requiredDuringSchedulingIgnoredDuringExecution affinity. A node must match the required rules before a pod can be scheduled to the node. ■ **Namespace**: Specify a namespace rule. Pod affinity is scoped to namespaces because it is enforced based on the labels of pods ■ **Topological Domain**: Set the topologyKey. This specifies the key for the node label that the system uses to denote the topological domain. For example, if you set the parameter to **kubernetes.io/hostname**, topologies are determined by nodes. If you set the parameter to **beta.kubernetes.io/os**, topologies are determined by the operating systems of nodes. ■ **Selector**: Click Add to add pod labels. ■ **View Applications**: Click **View Applications** and set the namespace and applications in the dialog box that appears. You can view the pod labels on the selected application and add the labels as selectors. ■ Required Rules: Specify labels on existing applications, operators, and label values. In this example, the required rule specifies that the application to be created is scheduled to a host that runs applications with the **app:nginx** label. ■ **Preferred**: Specify the rules that are preferred to match. Preferred rules correspond to the preferredDuringSchedulingIgnoredDuringExecution affinity. The scheduler attempts to schedule a pod to a node that matches the preferred rules. You can set node weights in preferred rules. Set the other parameters as described in the preceding settings. **Note** **Weight**: Set the weight of a preferred rule to a value from 1 to 100. The scheduler calculates the weight of each node that meets the preferred rule, and then schedules the pod to the node with the highest weight.

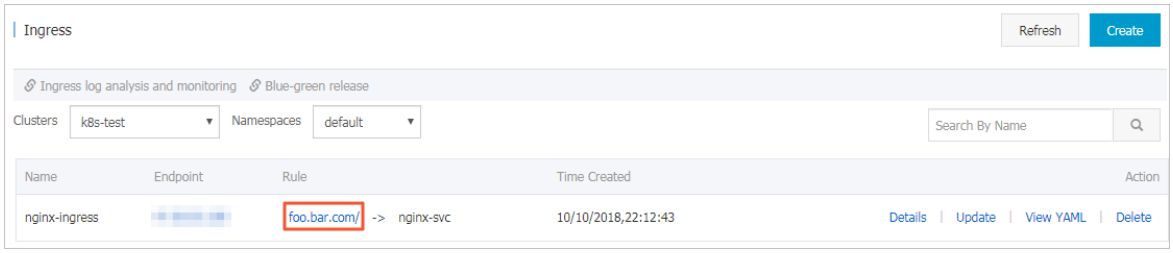
Parameter	Description
Pod Anti Affinity	Pod anti-affinity specifies that a pod is not scheduled to a node in the same topological domain if the node runs a pod that matches the anti-affinity rules. Pod anti-affinity rules apply to the following scenarios: ■ Schedule the pods of a Service to different topological domains, such as multiple hosts. This allows you to enhance the stability of the Service. ■ Grant a pod exclusive access to a node. This enables resource isolation and ensures that no other pod can share the resources of the specified node. ■ Schedule pods of Services to different hosts if these Services may interfere each other. **Note** You can set pod anti-affinity rules in the same way as setting pod affinity rules, and choose anti-affinity or pod affinity as needed.
Toleration	Set toleration rules to allow pods to be scheduled to nodes with matching taints.
Schedule to Virtual Nodes	Specify whether to schedule pods to virtual nodes. This option is unavailable if the cluster does not contain a virtual node.

- Labels and Annotations
  - Pod Labels: Add a label to the pod. The label is used to identify the application.
  - Pod Annotations: Add an annotation to the pod.

10. Click **Create**.
    After the application is deployed, you are redirected to the Complete page. You can find the resource objects under the application and click **View Details** to view application details.



11. Go to the details page of the cluster. In the left-side navigation pane, click **Ingresses**.
    You can find that the newly created Ingress rule appears on the page.



Result

Enter the test domain into the address bar of your browser and press the Enter key. The NGINX welcome page appears.

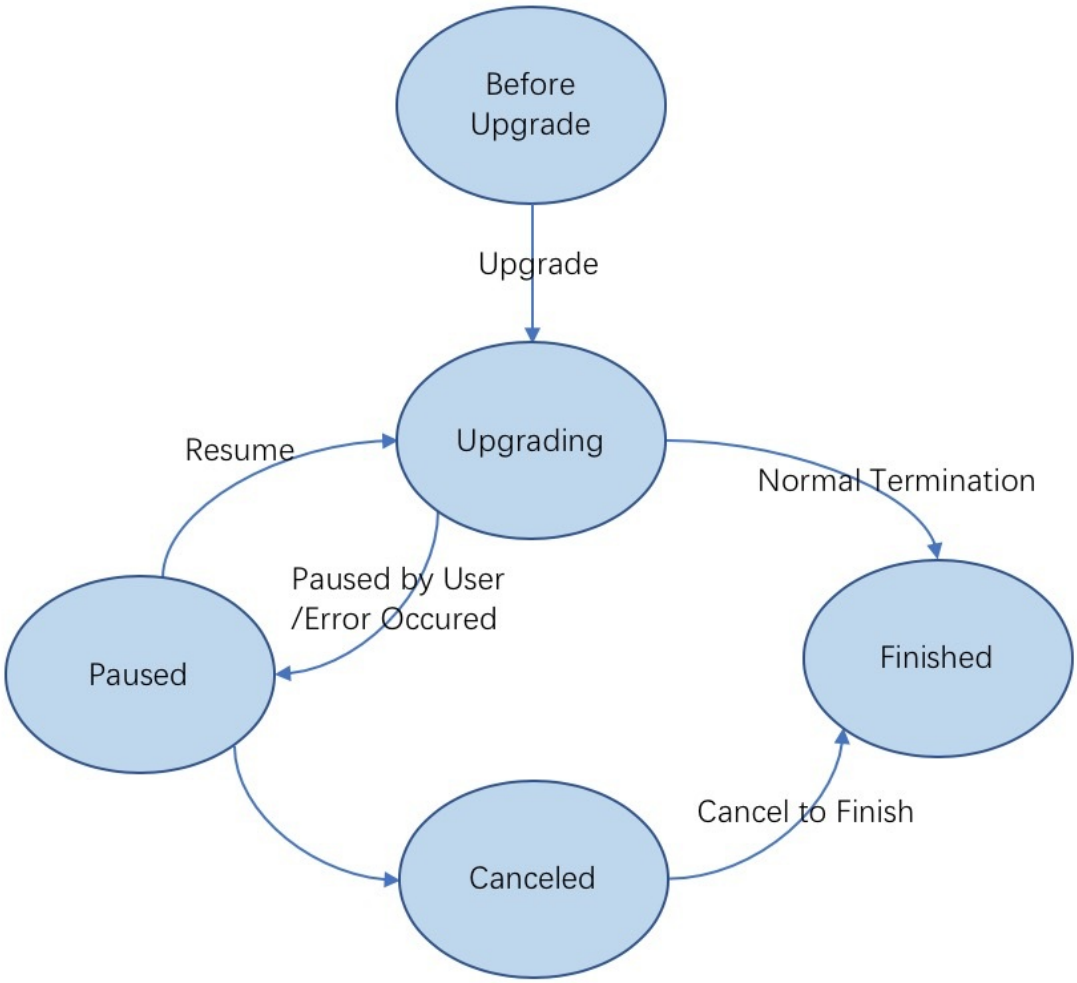
25.8. Upgrade the Sandboxed-Container runtime

You can upgrade the Sandboxed-Container runtime that is deployed on your nodes in the Container Service for Kubernetes (ACK) console. This topic describes how the Sandboxed-Container runtime is upgraded and how to perform an upgrade. It also lists the considerations to which you must pay attention during the upgrade.

How the Sandboxed-Container runtime is upgraded

- Upgrade process and status transitions

After you start an upgrade, the system automatically creates and executes upgrade tasks. The tasks are automatically divided into batches, distributed to nodes that require upgrades, and executed by running pods. During the upgrade process, you can pause, resume, or cancel the upgrade based on your business requirements.



- After you click **Upgrade**, the task changes to the **Upgrading** state.
- You can pause an **upgrading** task. After an upgrading task is paused, the task changes to the **Paused** state.
- You can resume a **paused** task. After a paused task is resumed, the task changes to the **Upgrading** state.
- You can also cancel a **paused** task. After a paused task is canceled, the task changes to the **Canceled** state.

- Upgrade policy
  Runtime upgrade is performed in batches:
  - The first batch includes one node. In subsequent batches, the number of nodes is increased by the power of 2. If you resume a paused upgrade, the first batch after the pause includes one node. The number of nodes is increased by the power of 2 in batches thereafter.
  - The maximum number of nodes in a batch does not exceed 10% of the total number of nodes.

  Assume that a cluster has 50 nodes that require upgrades. The upgrade process is proceeded in the following batches:

- The first batch includes one node.
- The second batch includes two nodes.
- The third batch includes four nodes.
- The fourth batch includes five nodes. Based on the calculation method, the number of nodes in the fourth batch is 8 (the ^{third} power of 2 is 8). However, the maximum number of nodes in a batch cannot exceed 10% of the total number of nodes. In this example, the maximum number = 50 × 10% = 5.
- The fifth batch includes five nodes.
- The following batches upgrade nodes in this way until all nodes are upgraded.

- Pause an upgrade
  You can pause an upgrade process at any time. Take the following notes when you pause an upgrade:
  - After you pause the upgrade, the upgrade will be completed on nodes where the upgrade has already started. The upgrade will not be performed on nodes where the upgrade has not started.
  - We recommend that you resume and complete a paused upgrade at your earliest convenience. Do not perform operations on the cluster when the upgrade is paused.
  - When you pause the upgrade, the system automatically checks whether nodes that have not started the upgrade exist. If no such nodes exist, you fail to pause the upgrade.

  After the upgrade is paused, you can click **Continue** to resume the upgrade.
  If an error occurs during the upgrade, the system automatically pauses the upgrade process. The cause of the error will be displayed at the bottom of the page. You can troubleshoot the error or Submit a ticket to request technical support.
- Cancel an upgrade
  After an upgrade is paused, you can click **Cancel** to cancel the upgrade. Take the following notes when you cancel an upgrade:
  - After you cancel the upgrade, the upgrade will be completed on nodes where the upgrade has already started. The upgrade will not be started on nodes where the upgrade has not started.
  - You cannot cancel the upgrade on the nodes where the upgrade is complete.

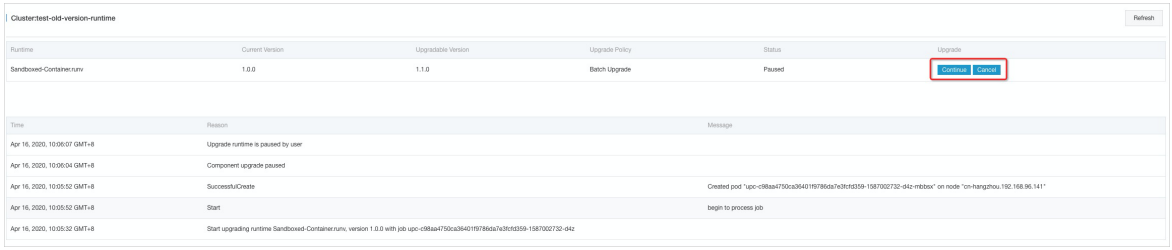
Notes

- To upgrade the runtime, nodes must have Internet access so that they can download upgrade packages.
- Failures may occur during the upgrade. To ensure data security, we recommend that you take snapshots of storage volumes before the upgrade.
- Applications that run in the cluster are not interrupted during the upgrade. We recommend that you check the release notes of the runtime to decide whether you need to release applications again. For more information, see Release notes of Sandboxed-Container.
- The upgrade is performed in batches. You can pause the upgrade after a batch has been upgraded. We recommend that you resume and complete a paused upgrade at your earliest convenience. Do not perform operations on the cluster when the upgrade is paused. If the upgrade has been paused for more than 15 days, it will be automatically canceled. The related events and log information are deleted.
- Do not add nodes to or remove nodes from the cluster during the upgrade. To add or remove nodes, you must first cancel the upgrade.
- Do not modify the resources in the **runtime-upgrade** namespace during the upgrade unless an error has occurred.
- If an error occurs during the upgrade, the process will be paused. You need to troubleshoot the error and delete the failed pods in the **runtime-upgrade** namespace. After the error is fixed, you can resume the upgrade.

> **Notice** Do not delete or modify resources other than the failed pods in the **runtime-upgrade** namespace even if an error occurs. You can contact the Alibaba Cloud technical support team for assistance.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the details page of the cluster, choose **Operations > Runtime Upgrade**.
5. On the **Runtime Upgrade** page, click **Upgrade**.
6. In the message that appears, click **OK**.
   To pause the upgrade, click **Pause**. After the upgrade is paused, you can click **Continue** to resume the upgrade or click **Cancel** to cancel the upgrade.



After you click **Upgrade** or **Continue**, you can view the operation records in the events list.

Result

After the upgrade is complete, the upgraded runtime no longer appears when you refresh or reopen the current page.

25.9. Security Sandbox configuration

25.9.1. Configure an ACK cluster that runs both sandboxed and Docker containers

This topic describes how to create a node pool of sandboxed containers and a node pool of Docker containers for a cluster of Container Service for Kubernetes (ACK). ACK allows you to create node pools of different container runtime types for a cluster. However, all nodes in a node pool must use the same type of container runtime.

Prerequisites

An ACK cluster is created. For more information, see 创建Kubernetes托管版集群.

> **Notice** The ACK cluster must meet the following requirements:
>
> - The ACK version must be 1.14.6 aliyun.1 or later.

- The network plug-in must be Flannel or Terway. Terway must run in One ENI for Multi-Pod mode.
- The volume plug-in must be CSI 1.14.8.39-0d749258-aliyun or later. You cannot use Flexvolume.
- The Logtail version must be 0.16.34.2-f6647154-aliyun or later.

Notes

- By default, you can deploy up to 100 nodes in an ACK cluster. To increase the quota of nodes, submit a ticket.
- Before you add an existing Elastic Compute Service (ECS) instance deployed in a virtual private cloud (VPC), make sure that an elastic IP address (EIP) is attached to the ECS instance, or a Network Address Translation (NAT) gateway is created in the same VPC. In addition, you must make sure that the related node can access the public network. Otherwise, you fail to add the ECS instance.

Create a node pool that runs Docker containers

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster in which you want to create a node pool and click **Node Pools** in the **Actions** column.
4. On the **Node Pools** page, click **Create Node Pool** and set the parameters.

   For more information, see 创建Kubernetes托管版集群. The following table lists the parameters.

Parameter	Description
Container Runtime	Select **Docker**. This specifies that all containers in the node pool are Docker containers.
Quantity	Specify the initial number of nodes in the node pool. If you do not need to create nodes, select 0.
Operating System	Select an operating system for the nodes. Valid values: CentOS, AliyunLinux, and Windows.

5. Click **OK**.

Create a node pool that runs sandboxed containers

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster in which you want to create a node pool and click **Node Pools** in the **Actions** column.
4. On the Node Pools page, click **Create Node Pool** and set the parameters.

   For more information, see 创建Kubernetes托管版集群. The following table lists the parameters.

Parameter	Description
Container Runtime	Select **Sandboxed-Container**. This specifies that all containers in the node pool are sandboxed containers.

Parameter	Description
Quantity	Specify the initial number of nodes in the node pool. If you do not need to create nodes, select 0.
Billing Method	Select Subscription. Sandboxed containers run only on ECS Bare Metal instances and ECS Bare Metal instances are billed on a subscription basis.
Mount Data Disk	You must mount a data disk of at least 200 GiB.
Operating System	Select AliyunLinux. Sandboxed containers support only the AliyunLinux operating system.

5. Click **OK**.

Result

- Check the status of the created node pool on the **Node Pools** page. If the status of the created node pool displays **Activated**, the node pool is created.
- To view detailed information about the nodes in the node pool, connect to the ACK cluster where the node pool is deployed.
  i. On the **Node Pools** page, click the name of the newly created node pool. In the **Node Pool Information** section, find and record the **ID of the node pool**.

  ii. Connect to the ACK cluster where the node pool is deployed. For more information, see Connect to Kubernetes clusters by using kubectl.
  iii. Run the following command to query the name of about a specified node:

```
kubectl get node --show-labels | grep -E "${node pool ID}|${node pool ID}"
```

  iv. Run the following command to query detailed information about a specified node:

```
kubectl get node -o wide | grep -E "${node name} {node name}"
```

25.9.2. Set kernel parameters for a sandboxed pod

Typically, the default kernel parameters provided by the containers in a pod can meet the requirements of most scenarios. However, in some scenarios, optimized kernel parameters may be required for better application performance. The Sandbox-Container runtime allows you to use annotations to set custom kernel parameters for a pod. You can also use the sysctl interface to set kernel parameters for containers that are in privileged mode. This topic describes how to set kernel parameters for a sandboxed pod.

Prerequisites

- Create a managed Kubernetes cluster that runs sandboxed containers.
- Connect to Kubernetes clusters by using kubectl.

Context

In a Kubernetes cluster, you can configure a security context and annotations for a pod to define safe and unsafe namespaced sysctls. You can also start a privileged container and configure unnamespaced sysctls.

Pods that use the runC runtime share kernels with the hosts. In this case, the following issues may arise:

- Privileged containers have almost full access permissions as the hosts. However, if you modify sysctls or unsafe sysctls, other pods may be affected or the nodes may become unavailable.
- You cannot set unnamespaced kernel parameters for different pods that are deployed on the same node.

Features of Sandbox-Container

A sandboxed pod has an exclusive kernel. You can use annotations to set custom kernel parameters for a pod or set containers to run in privileged mode. This addresses the issues caused by kernel sharing between hosts and pods that use the runC runtime.

- You can use annotations to set valid sysctls including namespaced and unnamespaced sysctls for a pod. The sysctls apply to only the kernel used by the current pod. Other pods or host sysctls are not affected.
- You can set a container to run in privileged mode. This grants the container almost full access permissions on the sandbox kernel. A sandbox has an exclusive kernel. The access permissions of the container are valid only within the sandbox. Other pods or host kernels are not affected.

Use annotations to configure sysctls for a pod

You can use the `securecontainer.alibabacloud.com/sysctls` annotation to configure sysctls for a pod. You can set one or more sysctls. Separate multiple sysctls with commas (,).

For example, if you want to enable packet forwarding, use the following annotation to configure sysctls:

```
annotations:
 securecontainer.alibabacloud.com/sysctls: "net.bridge.bridge-nf-call-ip6tables=1,net.bridge.bridge-nf-call-iptables=1,net.ipv4.ip_forward=1"
```

The following file is an example of a complete YAML file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
  app: busybox
 name: busybox
 namespace: default
spec:
 replicas: 2
 selector:
  matchLabels:
   app: busybox
 template:
  metadata:
   labels:
    app: busybox
   annotations:
    securecontainer.alibabacloud.com/sysctls: "net.bridge.bridge-nf-call-ip6tables=1,net.bridge.bridge-nf-call-iptables=1,net.ipv4.ip_forward=1"
  spec:
   containers:
    - command:
      - sleep
      - infinity
     image: 'busybox:latest'
     imagePullPolicy: IfNotPresent
     name: busybox
     resources:
      limits:
       cpu: '1'
       memory: 1Gi
      requests:
       cpu: 500m
       memory: 512Mi
   dnsPolicy: ClusterFirst
   restartPolicy: Always
   runtimeClassName: runv
```

Configure a privileged container

You can use the same method to provision privileged containers in a sandboxed pod and a pod that uses the runC runtime. However, you must configure the kernel host for a pod that uses the runC runtime but configure the sandbox kernel for a sandboxed pod.

You can run the following command to configure a privileged container:

```
containers:
 - name: busybox
   securityContext:
    privileged: true
```

The following file is an example of a complete YAML file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
  app: busybox
 name: busybox
 namespace: default
spec:
 replicas: 2
 selector:
  matchLabels:
   app: busybox
 template:
  metadata:
   labels:
    app: busybox
  spec:
   containers:
    - command:
      - sh
      - -c
      - "sysctl -w fs.inotify.max_user_watches=524286 && tail -f /dev/null"
     image: 'busybox:latest'
     imagePullPolicy: IfNotPresent
     name: busybox
     securityContext:
      privileged: true
     resources:
      limits:
       cpu: '1'
       memory: 1Gi
      requests:
       cpu: 500m
       memory: 512Mi
   dnsPolicy: ClusterFirst
   restartPolicy: Always
   runtimeClassName: runv
```

Related information

- Create a managed Kubernetes cluster that runs sandboxed containers
- Differences between runC and runV
- Connect to Kubernetes clusters by using kubectl

25.10. Security Sandbox storage

25.10.1. Mount a NAS file system to a sandboxed container

You can mount a Network Attached Storage (NAS) file system to a sandboxed container to significantly improve I/O performance. This topic describes how to mount a NAS file system to a sandboxed container.

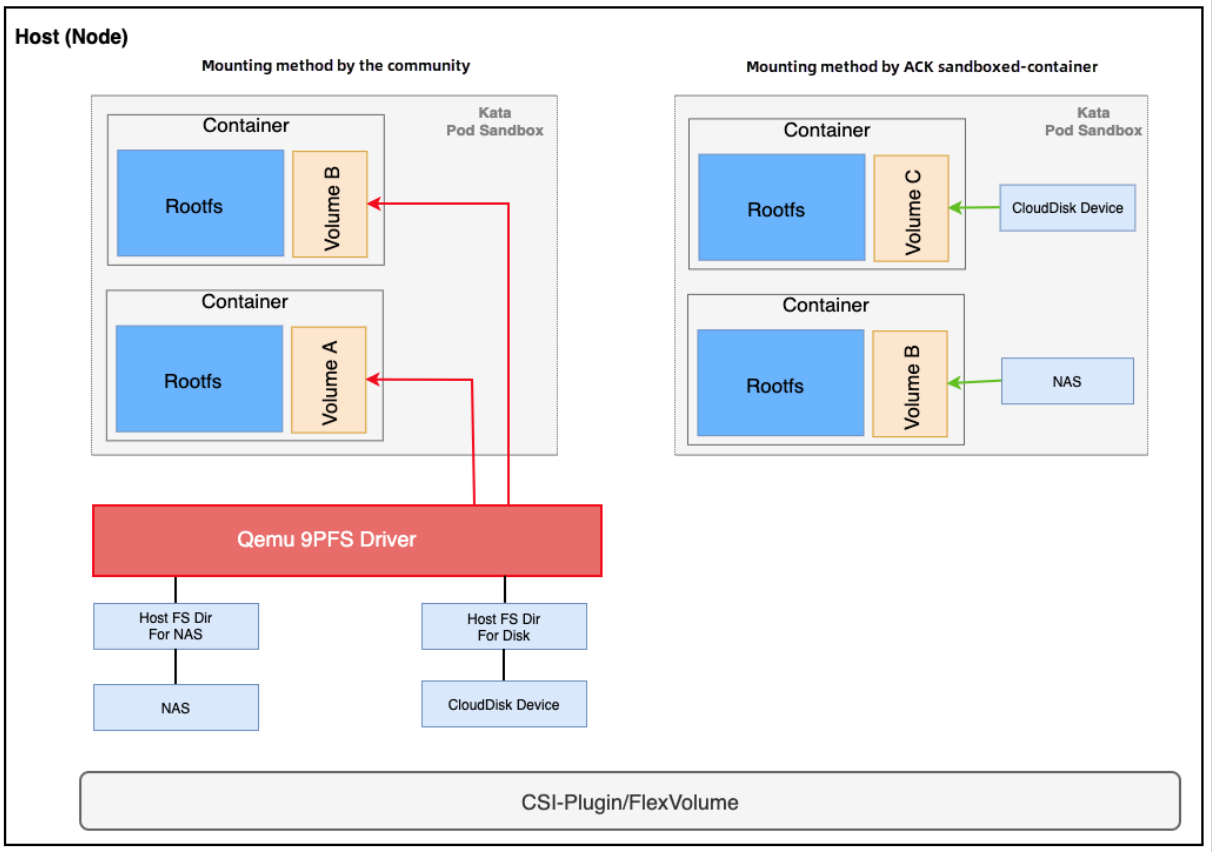
Prerequisites

- Create a managed Kubernetes cluster that runs sandboxed containers
- Connect to Kubernetes clusters by using kubectl

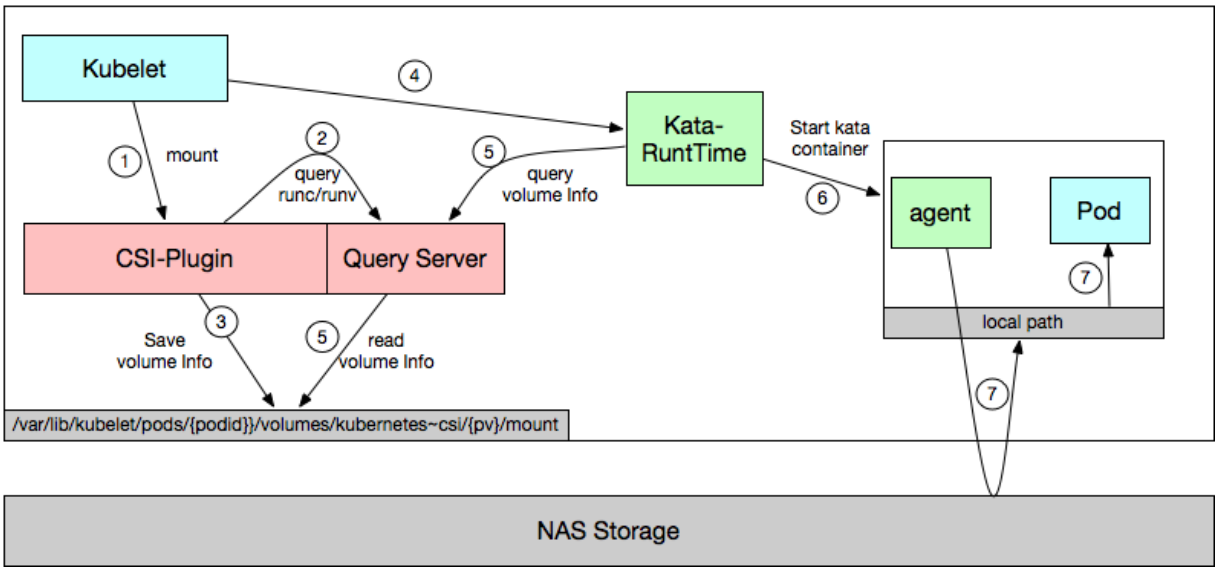
Context

virtio-fs is a shared file system. Container Service for Kubernetes (ACK) allows you to use virtio-fs to add volumes, Secrets, and ConfiMaps to the guest operating system of a virtual machine (VM). This directly mounts a NAS file system to a cluster. This method mounts the NAS file system to the host. Applications in the container can write data to and read data from the NAS file system only through virtio-fs. This may cause performance degradation.

Sandboxed containers allow you to directly mount NAS file systems. This method first unmounts NAS mount targets from the host. The NAS file system is mounted to the guest operating system. Then, the system creates a bind mount for the NAS file system. This way, applications in the container can directly write data to and read data from the NAS file system without performance degradation.



How a NAS file system is mounted to a sandboxed container



A NAS file system is mounted to a sandboxed container in the following process.

Step	Description
①	The kubelet requests the CSI plug-in to mount a NAS file system.
②	The CSI plug-in mounts the NAS file system to the host.
③	The Kubelet requests Kangaroo-Runtime to create a pod.
④	Kangaroo-Runtime parses the unmounting information, passes the information to the guest operating system, and then unmounts the NAS file system from the host.
⑤	Kangaroo-Runtime requests the agent to create a container.
⑥	The agent mounts the NAS file system to the guest operating system.
⑦	The agent creates a bind mount to mount for the NAS file system that is mounted to the guest operating system.

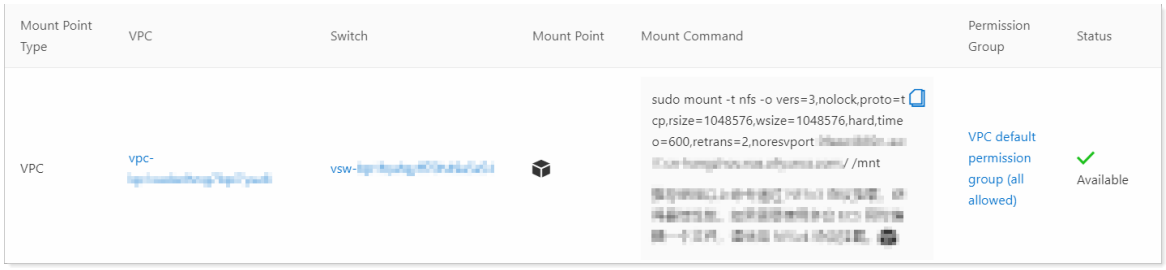
Examples

The following example describes how to mount a NAS file system to a sandboxed container. In this example, an Apsara File Storage NAS instance is created and a YAML file template is used to create resource objects.

1. Create an Apsara File Storage NAS instance. For more information, see Create a General-purpose NAS file system in the NAS console.

   > **Notice** The NAS file system must be deployed in the same virtual private cloud (VPC) as the cluster.

   Obtain the mount target of the NAS file system. As shown in the following figure, the URL *file-system-id .region.nas.aliyuncs.com* in the **Mount Command** column is the mount target.



Note *file-system-id.region.nas.aliyuncs.com* is the address of the mount target. You can obtain the address of the mount target in the NAS console. Log on to the NAS console, click the name of the NAS file system. In the left-side navigation pane, click **Mounting Usage**. You can find the address of the mount target in the Mount Command column.

2. Use the following template to create resource objects:

```
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: nas-pvc-csi
 namespace: default
spec:
 accessModes:
  - ReadWriteMany
 resources:
  requests:
   storage: 5Gi
 selector:
  matchLabels:
   alicloud-pvname: nas-pv-csi
---
apiVersion: v1
kind: PersistentVolume
metadata:
 labels:
  alicloud-pvname: nas-pv-csi
 name: nas-pv-csi
spec:
 accessModes:
  - ReadWriteMany
 capacity:
  storage: 5Gi
 csi:
  driver: nasplugin.csi.alibabacloud.com
  volumeAttributes:
   options: noresvport,nolock
   path: /csi
   server: ${nas-server-address}
   vers: "3"
  volumeHandle: nas-pv-csi
 persistentVolumeReclaimPolicy: Retain
---
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: deploy-nas-csi
spec:
  replicas: 2
  selector:
    matchLabels:
      app: busybox
  template:
    metadata:
      labels:
        app: busybox
      annotations:
        storage.alibabacloud.com/enable_nas_passthrough: "true"
    spec:
      runtimeClassName: runv
      containers:
        - name: busybox
          image: registry.cn-hangzhou.aliyuncs.com/acs/busybox:v1.29.2
          command:
          - tail
          - -f
          - /dev/null
          volumeMounts:
            - name: nas-pvc
              mountPath: "/data"
      restartPolicy: Always
      volumes:
        - name: nas-pvc
          persistentVolumeClaim:
            claimName: nas-pvc-csi
EOF
```

Note

- Replace `${nas-server-address}` in the template with the address of the mount target.

  ```
  server: ${nas-server-address}
  ```

- By default, pods cannot communicate with NAS file systems. You must add an annotation in the template to enable the pod to communicate with the NAS file system.

  ```
  annotations:
      storage.alibabacloud.com/enable_nas_passthrough: "true"
  ```

3. Run the following commands to query the ID of the pod and query the type of the file system that is mounted to the pod:

   ```
   kubectl get pods
   ```

   ```
   kubectl exec -it ${podid} sh
   ```

   ```
   mount | grep /data | grep nfs
   ```

   If the output is not empty, it indicates that the NAS file system is mounted to the container.

25.10.2. Mount a disk to a sandboxed container

You can mount a disk to a sandboxed container to significantly improve I/O performance. This topic describes how to mount a disk to a sandboxed container.

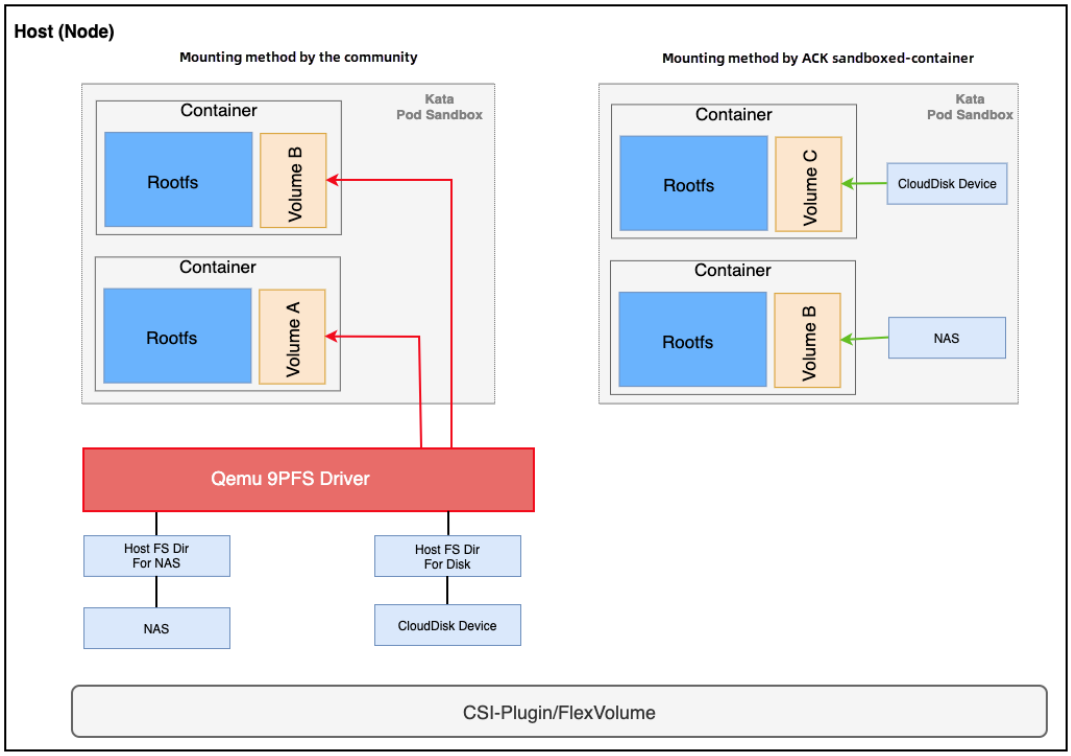
Prerequisites

- Create a managed Kubernetes cluster that runs sandboxed containers
- Connect to Kubernetes clusters by using kubectl

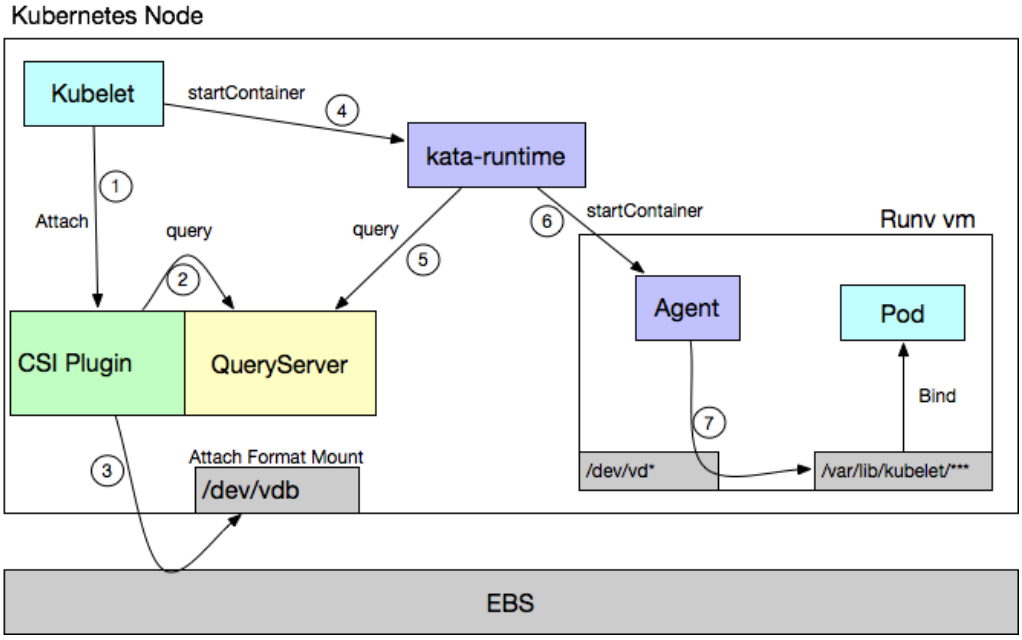
Context

virtio-fs is a shared file system. The Sandboxed-Container runtime provided by Container Service for Kubernetes supports virtio-fs. It allows you to add volumes, Secrets, and ConfigMaps to the guest operating system of a virtual machine (VM). This allows you to mount a disk as a volume to a sandboxed container. This method mounts the disk to the host. Applications in the container can write data to and read data from the disks only through virtio-fs. This may cause performance degradation.

Sandboxed containers allow you to directly mount disks. This method first unmounts disk mount targets from the host. Then, the disk is mounted to the guest operating system before the system creates a bind mount for the disk. This way, applications in the container can directly write data to and read data from the disk without performance degradation.



How a disk is mounted to a sandboxed container



A disk is mounted to a sandboxed container in the following process.

Step	Description
①	The kubelet requests the CSI plug-in to mount a disk.
②	The CSI plug-in formats the disk and mounts the disk to the host.
③	The kubelet requests Kangaroo-Runtime to create a pod.
④	Kangaroo-Runtime parses the disk unmounting information and unmounts the disk from the host.
⑤	Kangaroo-Runtime requests the agent to create a pod.
⑥	The agent mounts the disk to the guest operating system.
⑦	The agent creates a bind mount for the disk that is mounted to the guest operating system.

Examples

The following example shows how to mount a disk to a sandboxed container. In this example, a YAML file template is used to create resource objects.

1. Use the following template to create resource objects:

```
cat <<EOF | kubectl create -f -
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: alicloud-disk-ssd
parameters:
 type: cloud_ssd
provisioner: diskplugin.csi.alibabacloud.com
reclaimPolicy: Delete
---
apiVersion: v1
kind: PersistentVolumeClaim
```

```
kind: PersistentVolumeClaim
metadata:
 name: disk-pvc-01
 namespace: default
spec:
 accessModes:
  - ReadWriteOnce
 resources:
  requests:
   storage: 25Gi
 storageClassName: alicloud-disk-ssd
---
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
  app: busybox
 name: busybox
 namespace: default
spec:
 replicas: 1
 selector:
  matchLabels:
   app: busybox
 template:
  metadata:
   labels:
    app: busybox
   annotations:
    storage.alibabacloud.com/enable_ebs_passthrough: "true"
  spec:
   containers:
    - name: busybox
      image: registry.cn-hangzhou.aliyuncs.com/acs/busybox:v1.29.2
      command:
      - tail
      - -f
      - /dev/null
      volumeMounts:
       - mountPath: "/data"
         name: disk-pvc
   dnsPolicy: ClusterFirst
   restartPolicy: Always
   runtimeClassName: runv
   volumes:
    - name: disk-pvc
      persistentVolumeClaim:
       claimName: disk-pvc-01
EOF
```

> **Note**
> By default, pods cannot communicate with disks. You must add an annotation in the template to enable the pod to communicate with disks.
>
> ```
> annotations:
>     storage.alibabacloud.com/enable_ebs_passthrough: "true"
> ```

2. Run the following commands to check the type of file system that is mounted to the pod:

```
kubectl get pods
```

```
kubectl exec -it ${podid} sh
```

```
mount | grep /data | grep -vi virtio_fs
```

If the type of file system is not virtio-fs, it indicates that the disk is mounted to the container.

25.11. Compatibility notes

This topic describes the pod fields that are supported by Sandboxed-Container. The aim is to help you make full use of the Sandboxed-Container runtime.

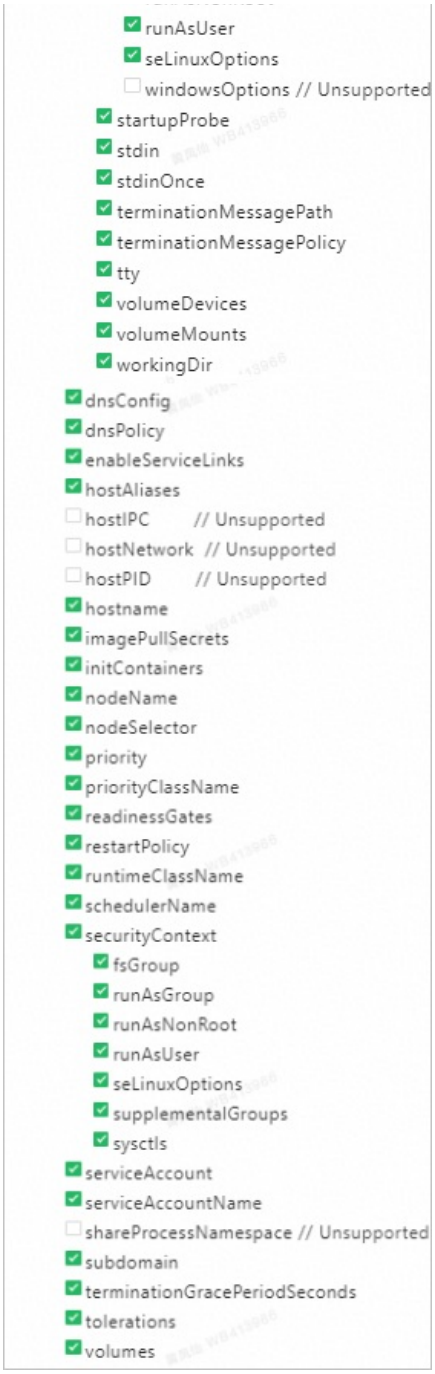
Context

Sandboxed-Container is a new runV container runtime that provides compatibility with runC in terms of pod networking, service networking (ClusterIP and NodePort), and image management. However, Sandboxed-Container does not support all pod fields. To use Sandboxed-Container, you do not need to change your development mode or image packaging method.

Supported pod fields

Sandboxed-Container supports the following pod fields (marked by ticks):

- pod
  - spec
    - ☑ activeDeadlineSeconds
    - ☑ affinity
    - ☑ automountServiceAccountToken
    - ☑ containers
      - ☑ args
      - ☑ command
      - ☑ env
      - ☑ envFrom
      - ☑ image
      - ☑ imagePullPolicy
      - ☑ lifecycle
      - ☑ livenessProbe
      - ☑ name
      - ☑ ports
      - ☑ readinessProbe
      - ☑ resources
      - ☑ securityContext
        - ☑ allowPrivilegeEscalation
        - ☑ capabilities
        - ☐ privileged // Unsupported
        - ☑ procMount
        - ☑ readOnlyRootFilesystem
        - ☑ runAsGroup
        - ☑ runAsNonRoot


runAsUser
seLinuxOptions
windowsOptions // Unsupported
startupProbe
stdin
stdinOnce
terminationMessagePath
terminationMessagePolicy
tty
volumeDevices
volumeMounts
workingDir
dnsConfig
dnsPolicy
enableServiceLinks
hostAliases
hostIPC // Unsupported
hostNetwork // Unsupported
hostPID // Unsupported
hostname
imagePullSecrets
initContainers
nodeName
nodeSelector
priority
priorityClassName
readinessGates
restartPolicy
runtimeClassName
schedulerName
securityContext
fsGroup
runAsGroup
runAsNonRoot
runAsUser
seLinuxOptions
supplementalGroups
sysctls
serviceAccount
serviceAccountName
shareProcessNamespace // Unsupported
subdomain
terminationGracePeriodSeconds
tolerations
volumes

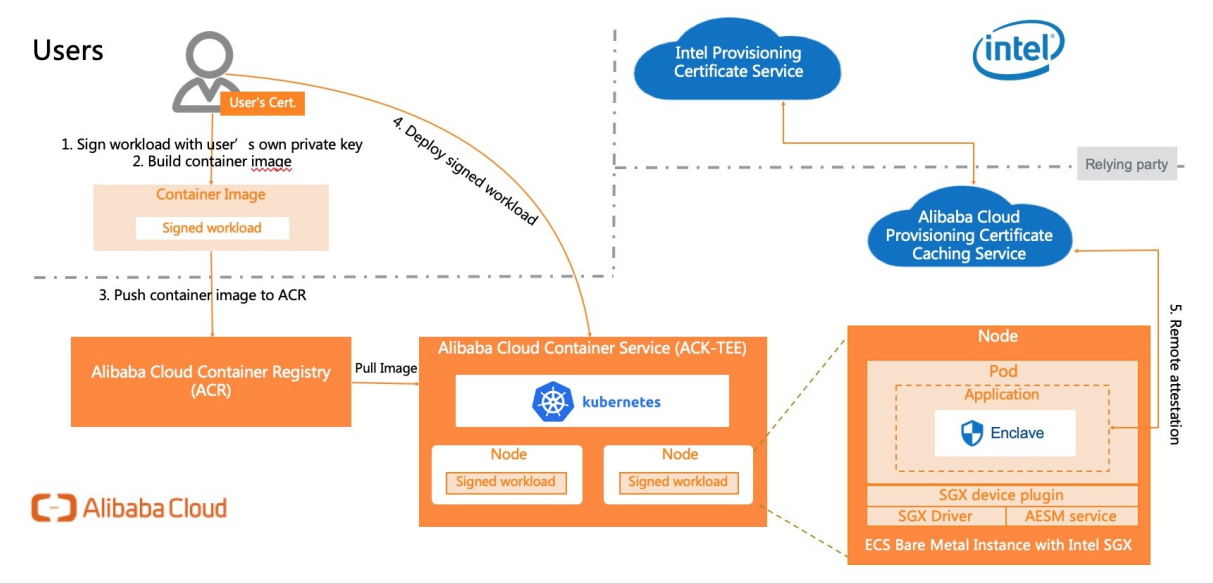
26.TEE-based confidential computing

26.1. TEE-based confidential computing

Container Service for Kubernetes clusters support confidential computing based on a trusted execution environment (TEE). This topic describes the purpose, features, scenarios, and solutions of TEE-based confidential computing. It also describes the collaboration between TEE-based confidential computing and sandboxed containers.

Concept

Container Service for Kubernetes provides TEE-based confidential computing. This is a cloud-native and all-in-one solution based on hardware encryption technologies. TEE-based confidential computing ensures the data security, integrity, and confidentiality. It also simplifies the development, delivery, and management costs of trusted or confidential applications. Confidential computing allows you to isolate sensitive data and code in a TEE. This prevents the rest part of the system from accessing the data and code. Encrypted data in the TEE is unavailable to other applications, the BIOS, operating systems, kernels, administrators, O&M engineers, cloud vendors, and hardware components except CPUs. This reduces the possibility of data breaches and simplifies data management.



Features

- Ensures the integrity of code and data in the cloud.
- Encrypts data and code to prevent data breaches.
- Enables lifecycle management of data.

Scenarios

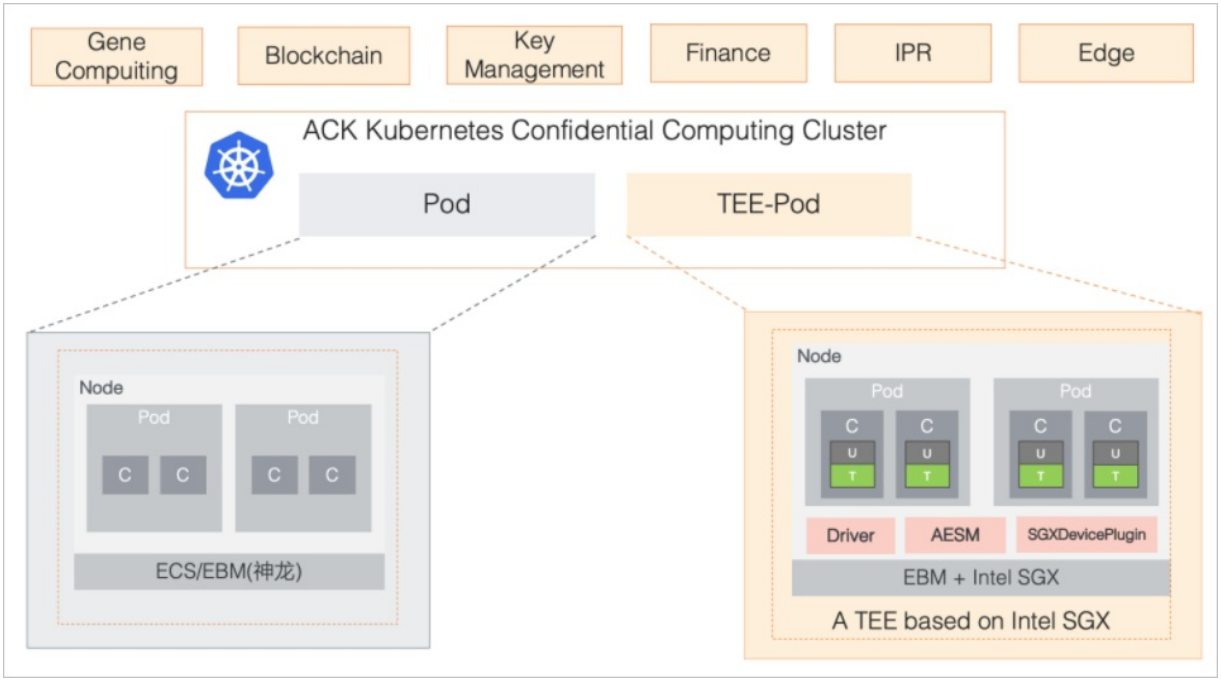
- **Blockchains**
  Enhances confidentiality and security for transaction processing, consensus, smart contracts, and key storage.
- **Key management**
  Deploys the key management feature in an enclave. This feature is similar to a hardware security module (HSM).
- **Genetic computing**

Ensures data confidentiality by isolating sensitive data in computing scenarios where multiple parties are involved.

- **Finance**
  Supports secure payments and transactions.
- **AI**
  Protects intellectual property rights by encrypting confidential information such as data models.
- **Edge computing**
  Supports secure and confidential communications among clouds, edges, and terminals.
- **Data sharing and computing**
  Protects data from breaches when users or vendors share data for higher economic value.

Solution

The following figure shows the TEE-based confidential computing v1.0.



Container Service for Kubernetes supports Intel Software Guard Extensions (SGX)-based confidential computing in a managed cluster. This feature simplifies management and delivery of trusted or confidential applications at reduced costs. Confidential computing ensures the integrity and confidentiality of data and code in public clouds and prevents access from cloud vendors. For more information about how to create a managed Kubernetes cluster for confidential computing, see Create a managed Kubernetes cluster for confidential computing.

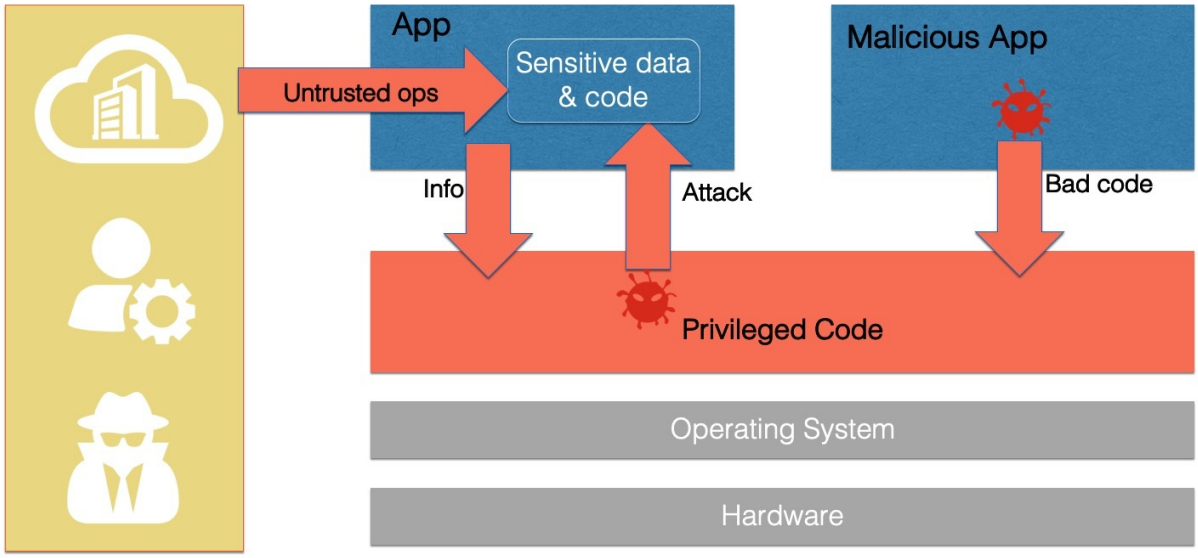
Make sure that the following requirements are met:

- Worker nodes must be ECS Bare Metal instances of the **ecs.ebmhfg5.2xlarge** type. This instance type supports Intel SGX.
- The SGX driver and SGX Platform Software (PSW) are automatically installed during node initialization.
- By default, Intel SGX Architectural Enclave Service Manager (AESM) DaemonSet is installed. This allows SGX applications to access AESM.
- The SGX device plug-in that is developed by Alibaba Cloud simplifies the detection, management, and scheduling of memory resources in Enclave Page Cache (EPC) of SGX nodes.

TEE-based confidential computing collaborates with sandboxed containers

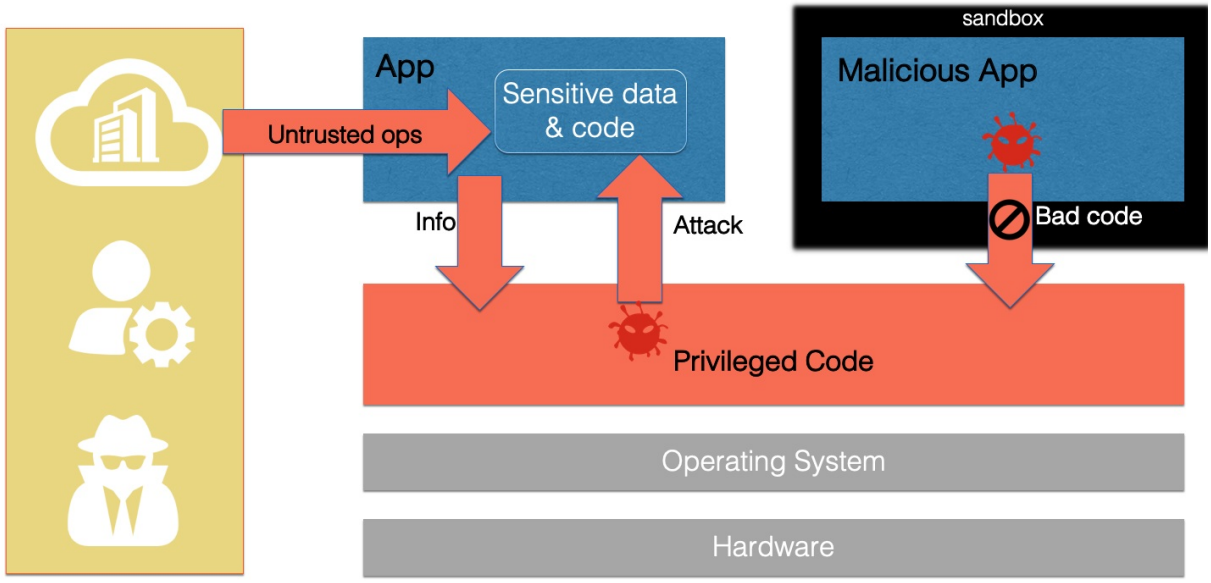
Containers in runC are vulnerable to attacks

A container in runC shares a kernel with the host. When a container escape vulnerability is detected in the kernel, malicious applications in the container may escape into the backend system. This may cause negative effects to other applications and the entire system.



Sandboxed containers isolate malicious applications and block attacks

Sandboxed containers provide enhanced isolation based on the lightweight Kangaroo framework. Each pod runs on an independent operating system and kernel. When a vulnerability is detected in a kernel, only the pod that runs on this kernel is affected. This protects other applications and the backend system.

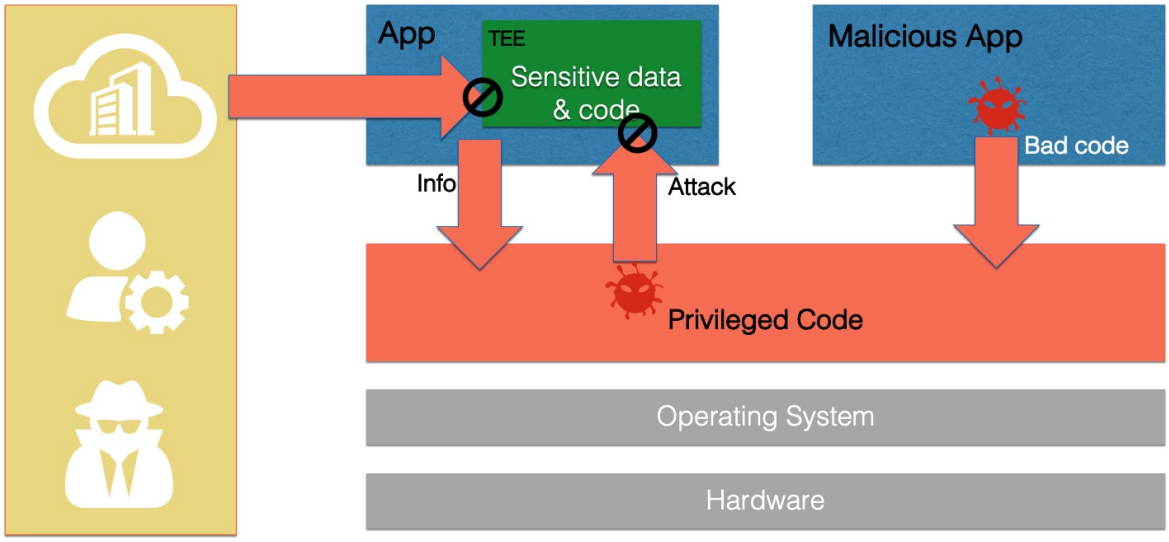


TEE-based confidential computing protects applications in use

TEE-based confidential computing is an encrypted computing solution provided by Container Service for Kubernetes. It protects sensitive code and data, such as IP addresses, keys, and confidential communications.

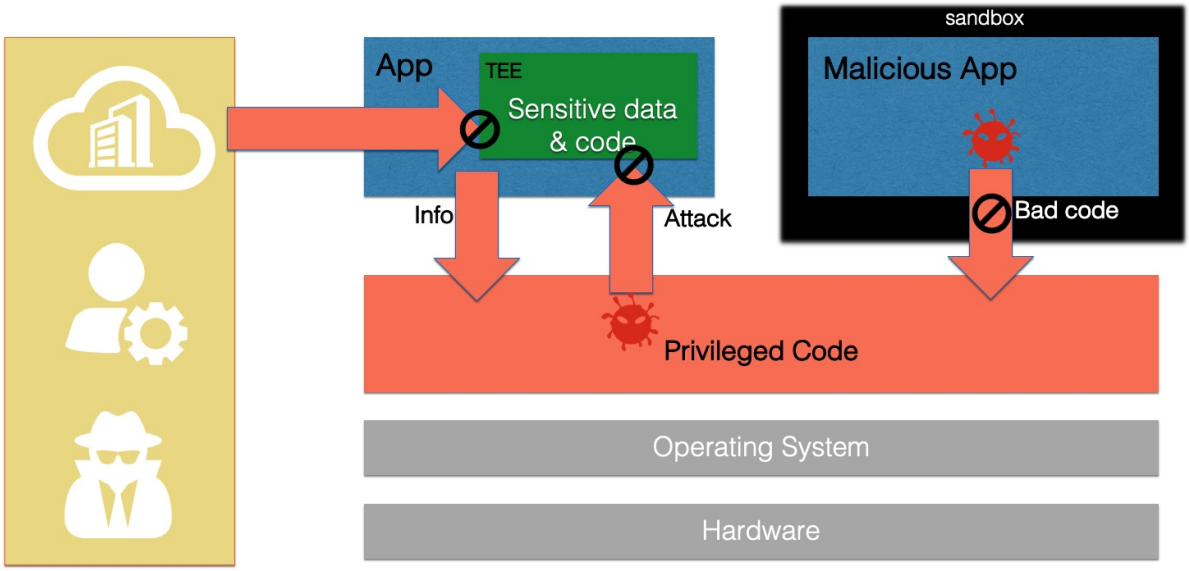
Cloud computing is a technology that brings benefits to enterprises. However, when you migrate data to the cloud, the possibility of data breaches becomes a core concern. Data breaches may occur in the following scenarios:

- Attacks
- Untrusted cloud vendors
- Security flaws of cloud infrastructures
- Unqualified O&M personnel and administrators



TEE-based confidential computing collaborates with sandboxed containers to isolate malicious applications and protect sensitive data

TEE-based confidential computing and sandboxed containers provide different features. You can combine the features in the cluster to isolate malicious applications and protect sensitive applications and data.



Related information

- Create a managed Kubernetes cluster for confidential computing
- Create a managed Kubernetes cluster that runs sandboxed containers

26.2. Create a managed Kubernetes cluster for confidential computing

This topic describes how to create a managed Kubernetes cluster for confidential computing in the Container Service for Kubernetes (ACK) console.

Prerequisites

ACK and Resource Access Management (RAM) are activated.

ACK is activated in the ACK console and RAM is activated in the RAM console.

Note

When you create a managed Kubernetes cluster for confidential computing, take note of the following limits:

- Server Load Balancer (SLB) instances that are created along with an ACK cluster support only the pay-as-you-go billing method.
- ACK clusters support only VPCs.
- By default, each account has specific quotas on cloud resources that can be created. You cannot create clusters if the quota is reached. Make sure that you have sufficient quotas before you create a cluster. To increase quotas,submit a ticket.
  - By default, you can create up to 50 clusters across all regions with each account. Each cluster can contain up to 100 nodes. To increase the quota of clusters or nodes,submit a ticket.
    By default, you can add up to 48 route entries to a virtual private cloud (VPC). This means that you can deploy up to 48 nodes in an ACK cluster that uses Flannel. An ACK cluster that uses Terway is not subject to this limit. To deploy more nodes in this case,submit a ticket to apply for an increase on the quota of route entries in the VPC that you want to use.
  - By default, you can create up to 100 security groups with each account.
  - By default, you can create up to 60 pay-as-you-go SLB instances with each account.
  - By default, you can create up to 20 elastic IP addresses (EIPs) with each account.
- To create a managed Kubernetes cluster for confidential computing, you must set the parameters as described in the following table. Otherwise, you cannot run Intel Software Guard Extensions (SGX) applications in the cluster.

Parameter	Description
Zone	Only Elastic Compute Service (ECS) instances of the **c7t security-enhanced compute optimized instance family** support clusters for confidential computing. Make sure that these instances are available in the selected zone.
Container Runtime	Select containerd 1.4.4 or later.

Parameter	Description
Instance Type	You must select ECS instances of the **c7t security-enhanced compute optimized instance family**. **Note** ◦ The **security-enhanced compute optimized** instance family is in private preview. This instance family provides a limited stock of ECS instances and is not covered by terms of service level agreement (SLA). If you want to use ECS instances of this instance family, apply for ECS instances based on your minimum requirement. ◦ Intel Ice Lake supports the remote attestation service only based on Intel Software Guard Extensions Data Center Attestation Primitives (SGX DCAP). Remote attestation services based on Intel Enhanced Privacy Identification (EPID) are not supported. You must adapt your applications before you can use the remote attestation service. For more information about the remote attestation service, see attestation-services.
Operating System	Select **AliyunLinux 2.xxxx 64-bit (UEFI)**
Network Plug-in	Select **Flannel**.

Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Cluster Template**.
4. In the **Select Cluster Template** dialog box, find **Confidential Computing Cluster** in the **Managed Clusters** section and click **Create**.
5. On the **Managed Kubernetes** tab, configure the cluster.
   i. Configure basic settings of the cluster.

Parameter	Description

Parameter	Description
Cluster Name	Enter a name for the ACK cluster. **Note** The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-).
Cluster Specification	Select a cluster type. You can select **Standard edition** or **Professional**.
Region	Select a region to deploy the cluster.
All Resources	Move the pointer over **All Resources** at the top of the page and select the resource group that you want to use. After you select a resource group, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource group. When you create a cluster, only the VPCs and vSwitches that belong to the selected resource group are displayed in the console. 
Kubernetes Version	The Kubernetes versions that are supported by ACK.
Confidential Computing	Select **Enable**.
Container Runtime	Only the containerd runtime is supported. For more information, see Comparison of Docker, containerd, and Sandboxed-Container.
VPC	Select a VPC to deploy the cluster. Standard VPCs and shared VPCs are supported. ■ Shared VPC: The owner of a VPC (resource owner) can share the vSwitches in the VPC with other accounts in the same organization. ■ Standard VPC: The owner of a VPC (resource owner) cannot share the vSwitches in the VPC with other accounts. **Note** ACK clusters support only VPCs. You can select a VPC from the drop-down list. If no VPC is available, click **Create VPC** to create one. For more information, see Work with VPCs.
VSwitch	Select vSwitches. You can select up to three vSwitches that are deployed in different **zones**. If no vSwitch is available, click **Create VSwitch** to create one. For more information, see Work with vSwitches.
Network Plug-in	You must select the **Flannel** plug-in if you want to enable confidential computing.

Parameter	Description
Pod CIDR Block	If you select **Flannel** as the network plug-in, you must set **Pod CIDR Block**. The CIDR block specified by **Pod CIDR Block** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
Service CIDR	Set **Service CIDR**. The CIDR block specified by **Service CIDR** cannot overlap with that of the VPC or those of the existing clusters in the VPC. The CIDR block cannot be modified after the cluster is created. The Service CIDR block cannot overlap with the pod CIDR block. For more information about subnetting for ACK clusters, see Plan CIDR blocks for an ACK cluster.
IP Addresses per Node	If you select **Flannel** as the network plug-in, you must set **IP Addresses per Node**. Note ■ **IP Addresses per Node** specifies the maximum number of IP addresses that can be assigned to each node. We recommend that you use the default value. ■ After you select the VPC and specify the number of IP addresses per node, recommended values are automatically generated for Pod CIDR block and Service CIDR block. The system also provides the maximum number of nodes that can be deployed in the cluster and the maximum number of pods that can be deployed on each node. You can modify the values based on your business requirements.
Configure SNAT	By default, an ACK cluster cannot access the Internet. If the VPC that you select for the ACK cluster cannot access the Internet, you can select **Configure SNAT for VPC**. This way, ACK will create a NAT gateway and configure Source Network Address Translation (SNAT) rules to enable Internet access for the VPC.

Parameter	Description
Access to API Server	By default, an internal-facing Server Load Balancer (SLB) instance is created for the cluster API server. You can modify the specification of the SLB instance. For more information, see Instance types and specifications. **Notice** If you delete the SLB instance, you cannot access the cluster API server. Select or clear **Expose API Server with EIP**. The ACK API server provides multiple HTTP-based RESTful APIs, which can be used to create, delete, modify, query, and monitor resources, such as pods and Services. ■ If you select this check box, an elastic IP address (EIP) is created and associated with an Internet-facing SLB instance. Port 6443 used by the API server is opened on master nodes. You can connect to and manage the ACK cluster by using kubeconfig over the Internet. ■ If you clear this check box, no EIP is created. You can connect to and manage the ACK cluster by using kubeconfig only within the VPC.
RDS Whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the cluster to the RDS whitelist. **Note** To enable an RDS instance to access the cluster, you must deploy the RDS instance in the VPC where the cluster is deployed.
Security Group	You can select **Create Basic Security Group**, **Create Advanced Security Group**, or **Select Existing Security Group**. For more information, see Overview. **Note** To select **Select Existing Security Group**, Submit a ticket to apply to be added to a whitelist.
Secret Encryption	Select or clear **Select Key**.

ii. Configure advanced settings of the cluster.

Parameter	Description
Kube-proxy Mode	iptables and IPVS are supported. ■ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the ACK cluster. This mode is suitable for ACK clusters that manage a small number of Services. ■ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for ACK clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.

Parameter	Description
Labels	Add labels to the cluster. Enter a key and a value, and click **Add**. **Note** ■ Key is required. *Value* is optional. ■ Keys are not case-sensitive. A key must be 1 to 64 characters in length, and cannot start with aliyun, http://, or https://. ■ *Values* are not case-sensitive. A value can be empty and can contain up to 128 characters in length. It cannot be http:// or https://. ■ The keys of labels that are added to the same resource must be unique. If you add a label with a used key, the label overwrites the others that use the same key. ■ If you add more than 20 labels to a resource, all labels become invalid. You must remove excess labels for the remaining labels to take effect.
Cluster Domain	Set the domain name of the cluster. **Note** The default domain name is **cluster.local**. You can enter a custom domain name. A domain name consists of two parts. Each part must be 1 to 63 characters in length and can contain only letters and digits. You cannot leave these parts empty.
Custom Certificate SANs	You can enter custom subject alternative names (SANs) for the API server certificate of the cluster to accept requests from specified IP addresses or domain names. For more information, see Customize the SAN of the API server certificate for a managed Kubernetes cluster.
Service Account Token Volume Projection	**Service account token volume projection** reduces security risks when pods use service accounts to access the API server. This feature enables kubelet to request and store the token on behalf of the pod. This feature also allows you to configure token properties, such as the audience and validity duration. For more information, see Enable service account token volume projection.
Deletion Protection	

6. Click **Next: Worker Configurations** to configure worker nodes.
   i. Set worker nodes.

Note You can select only ECS instances of the **c7t security-enhanced compute optimized instance family**. This enables the managed Kubernetes cluster for confidential computing to run applications of Intel SGX 2.0.

- If you select **Create Instance**, you must set the parameters that are listed in the following table.

Parameter	Description
Billing Method	The **pay-as-you-go** and **subscription** billing methods are supported. If you select the **subscription** billing method, you must set the following parameters: ■ **Duration**: You can select 1, 2, 3, or 6 months. If you require a longer duration, you can select 1 to 5 years. ■ **Auto Renewal**: Specify whether to enable auto-renewal. The following billing methods are supported: **Pay-As-You-Go** and **Subscription**.
Instance Type	Only ECS instances of the **c7t security-enhanced compute optimized instance family** can run applications of Intel SGX 2.0.
Selected Types	The selected instance types. You can select multiple instance types.
Quantity	Specify the number of worker nodes (ECS instances) to be created.
System Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. Note ■ You can select **Enable Backup** to back up disk data. ■ If you select **enhanced SSD** as the system disk type, you can set a custom **performance level** for the system disk. You can select higher performance levels for enhanced SSDs with larger storage capacities. For example, you can select performance level 2 for an enhanced SSD with a storage capacity of more than 460 GiB. You can select performance level 3 for an enhanced SSD with a storage capacity of more than 1,260 GiB. For more information, see Capacity and PLs.
Mount Data Disk	**Enhanced SSDs**, **standard SSDs**, and **ultra disks** are supported. You can enable **disk encryption** and **disk backup** when you mount a data disk.
Operating System	Only the AliyunLinux 2.xxxx 64-bit (UEFI) operating systems are supported.
Logon Type **Key Pair**	■ Key pair logon ■ **Key Pair**: Select an SSH key pair from the drop-down list. ■ **create a key pair**: Create an SSH key pair if none is available. For more information about how to create an SSH key pair, see Create an SSH key pair. After the key pair is created, set it as the credential that is used to log on to the cluster. ■ Password logon ■ **Password**: Enter the password that is used to log on to the nodes. ■ **Confirm Password**: Enter the password again.

- If you select **Add Existing Instance**, you must add at least two worker nodes. You must also set **Duration**, **Auto Renewal**, **Operating System**, **Logon Type**, and **Key Pair** based on the preceding settings.

ii. Configure advanced settings of the worker nodes.

Parameter	Description
Node Protection	Specify whether to enable node protection. **Note** By default, this check box is selected. Node protection prevents nodes from being accidentally deleted in the console or by calling the API. This prevents user errors.
User Data	You can enter custom scripts. Custom scripts are automatically executed after the nodes are initialized. **Note** Windows nodes support Batch and PowerShell scripts. Before you encode the content in Base64, make sure that the first line includes [bat] or [powershell]. Linux nodes support shell scripts. For more information about the supported formats, see cloud-init and Overview of ECS instance user data. If your script file is larger than 1 KB, we recommend that you upload the script to an Object Storage Service (OSS) bucket and pull the script from the internal endpoint of the OSS bucket.
Custom Image	**Notice** Do not use custom images. Some images may not support confidential computing.
Custom Node Name	Specify whether to use a **custom node name**. A node name consists of a prefix, an IP substring, and a suffix. ■ Both the prefix and suffix can contain one or more parts that are separated by periods (.).These parts can contain lowercase letters, digits, and hyphens (-), and must start and end with a lowercase letter or digit. ■ The IP substring length specifies the number of digits to be truncated from the end of the returned node IP address. Valid values: 5 to 12.
CPU Policy	Set the CPU policy. ■ none: This policy indicates that the default CPU affinity is used. This is the default policy. ■ static: This policy allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Taints	Add taints to the worker nodes in the ACK cluster.

7. Click **Next: Component Configurations** to configure components.

Parameter	Description

Parameter	Description
Ingress	Specify whether to install Ingress controllers. By default, **Install Ingress Controllers** is selected. For more information, see Ingress高级用法. **Note** If you want to select **Create Ingress Dashboard**, you must first enable Log Service.
Monitoring Agents	Select whether to install the CloudMonitor agent. After the CloudMonitor agent is installed on ECS instances, you can view monitoring information about the instances in the CloudMonitor console.
Log Service	Specify whether to enable Log Service. You can select an existing Log Service project or create one. By default, **Enable Log Service** is selected. When you create an application, you can enable Log Service through a few steps. For more information, see Collect log files from containers by using Log Service.
Workflow Engine	Specify whether to enable Alibaba Cloud Genomics Service (AGS). **Note** To use this feature, submit a ticket to apply to be added to a whitelist. ○ If you select this check box, the system automatically installs the AGS workflow plug-in when the system creates the cluster. ○ If you clear this check box, you must manually install the AGS workflow plug-in. For more information, see Introduction to AGS CLI.

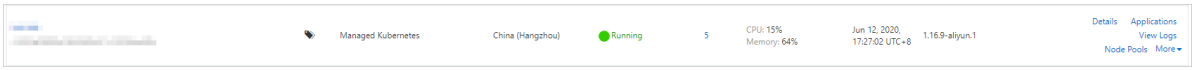
8. Click **Next: Confirm Order** to confirm the cluster configurations and select **Terms of Service**.
9. Click **Create Cluster** to deploy the cluster.

   You can also view detailed information about how the cluster is created in the log data.

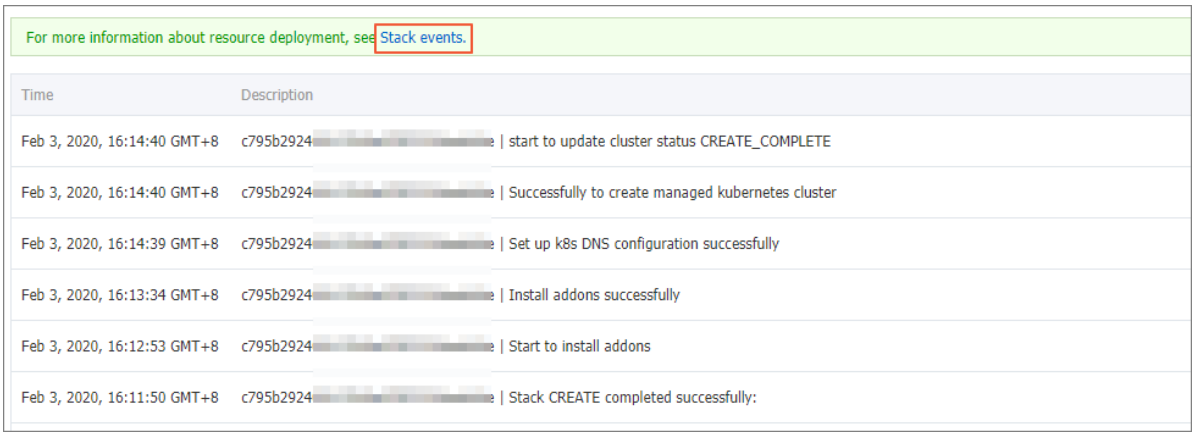
Note It requires about 10 minutes to create a managed Kubernetes cluster that consists of multiple nodes.

Result

- After the cluster is created, you can view the created cluster on the **Clusters** page in the console.



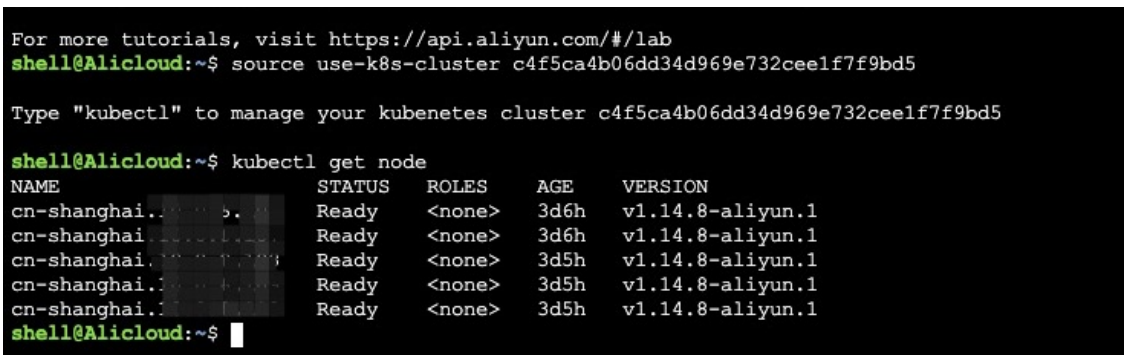
- Click **View Logs** in the **Actions** column. On the **Log Information** page, you can view the cluster log. To view detailed log information, click **Stack events**.



- On the **Clusters** page, find the created cluster, and click its name or click **Details** in the Actions column. On the details page, you can click the **Basic Information** tab to view basic information about the cluster and click the **Connection Information** tab to view information about how to connect to the cluster.
  The following information is displayed:
  - **API Server Public Endpoint**: the IP address and port that the API server uses to provide services over the Internet. It allows you to manage the cluster by using kubectl or other tools on your terminal.
    Bind EIP and **Unbind EIP**: These options are available to only managed Kubernetes clusters.
    - Bind EIP: You can select an existing elastic IP address (EIP) or create one.
      The API server restarts after you map an EIP to the API server. We recommend that you do not perform operations on the cluster during the restart process.
    - Unbind EIP: You cannot access the API server over the Internet after you remove mapping.
      The API server restarts after you remove mapping of the EIP from the API server. We recommend that you do not perform operations on the cluster during the restart process.
  - **API Server Internal Endpoint**: the IP address and port that the API server uses to provide services within the cluster. The IP address belongs to the Server Load Balancer (SLB) instance that is bound to the cluster.
  - **Testing Domain**: the domain name that is used to test Services. The suffix of the domain name is `<cluster_id>.<region_id>.alicontainer.com`.

    > **Note** To remap the domain name, click **Rebind Domain Name**.

- You can Connect to Kubernetes clusters by using kubectl and run the `kubectl get node` command to query information about the nodes in the cluster.



Related information

- TEE-based confidential computing
- Use TEE SDK to develop and build applications of Intel SGX 2.0

26.3. Create a node pool that supports confidential computing

You can create a node pool that supports confidential computing in a Container Service for Kubernetes (ACK) cluster to enable confidential computing. This node pool provides a Trusted Execution Environment (TEE) that can protect your code and sensitive data from being sniffed or compromised when the code or data is in use. This topic describes how to create a node pool that supports confidential computing.

Prerequisites

- A managed Kubernetes cluster is created. For more information, see 创建Kubernetes托管版集群. The created cluster must meet the following requirements:
  - The network plug-in is Flannel.
  - The container runtime is Docker.
- The cluster is deployed in a region where you can purchase ECS Bare Metal instances of the ecs.ebmhfg5.2xlarge type.

Context

TEE-based confidential computing for ACK is powered by Intel Software Guard Extensions (Intel SGX). It provides a cloud-native, all-in-one platform for you to develop and manage confidential computing applications. Only trusted applications are allowed to run within TEEs. This ensures the security, integrity, and confidentiality of the data that is in use. Confidential computing allows you to isolate sensitive data and code in a TEE. This prevents the data and code from being accessed by the rest of the system. The data stored within a TEE is inaccessible to external applications, the BIOS, the operating system, the kernel, administrators, O&M engineers, cloud service providers, and hardware components except the CPU. This reduces the possibility of data leakage and simplifies data management. You can create a node pool that supports confidential computing in a managed Kubernetes cluster to provide confidential computing for the cluster.

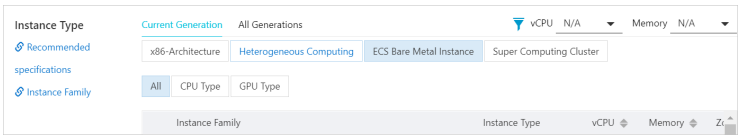
Procedure

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
4. In the upper-right corner of the **Node Pools** page, click **Create Node Pool**.

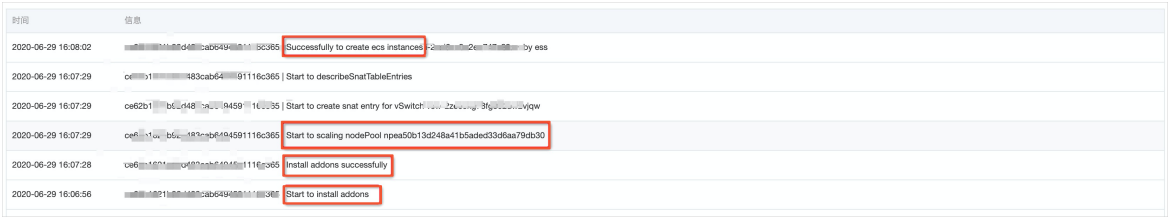
   In the upper-right corner of the **Node Pools** page, you can also click **Create Managed Node Pool** to create a managed node pool, or click **Configure Auto Scaling** to create an auto-scaling node pool.
5. In the **Create Node Pool** dialog box, set the parameters.

   For more information, see 创建Kubernetes托管版集群. The following table describes the parameters that are required to create a node pool that supports confidential computing.

Parameter	Description
Confidential Computing	Select **Enable** to enable confidential computing. 

Parameter	Description
Container Runtime	Only containerd and Docker support confidential computing. **Note**
Auto Scaling	Specify whether to enable auto scaling of the node pool. If you enable auto scaling, the node pool automatically scales based on resource consumption.
Instance Type	Select **ECS Bare Metal Instance** and select **ecs.ebmhfg5.2xlarge** as the **instance type**. **Note** You can select multiple instance types. However, only the **ecs.ebmhfg5.2xlarge** instance type supports confidential computing. If the stock of ecs.ebmhfg5.2xlarge instances is insufficient, you can select another instance type. However, the node pool does not support confidential computing in this case. 
Quantity	Specify the initial number of the nodes in the node pool. If you do not want to create nodes in the node pool, set this parameter to 0.
Operating System	You can select only the Aliyun Linux operating system.
Node Label	You can add labels to the nodes in the node pool.
ECS Label	You can add labels to the ECS instances in the node pool.

6. Click **Confirm Order**.
   On the **Node Pools** page, if the **state** of the node pool is **Initializing**, it indicates that the system is creating the node pool.
   On the **Clusters** page, find the cluster and click **View Logs** in the **Actions** column. On the Log Information page, you can view the logs of the newly created node pool that supports confidential computing.



   After the node pool is created, the **state** of the node pool changes to **Active**.

What's next

After the node pool that supports confidential computing is created, you can create and deploy Intel SGX

applications. For more information, see Use TEE SDK to develop and build applications of Intel SGX 2.0.

26.4. Use TEE SDK to develop and build applications of Intel SGX 2.0

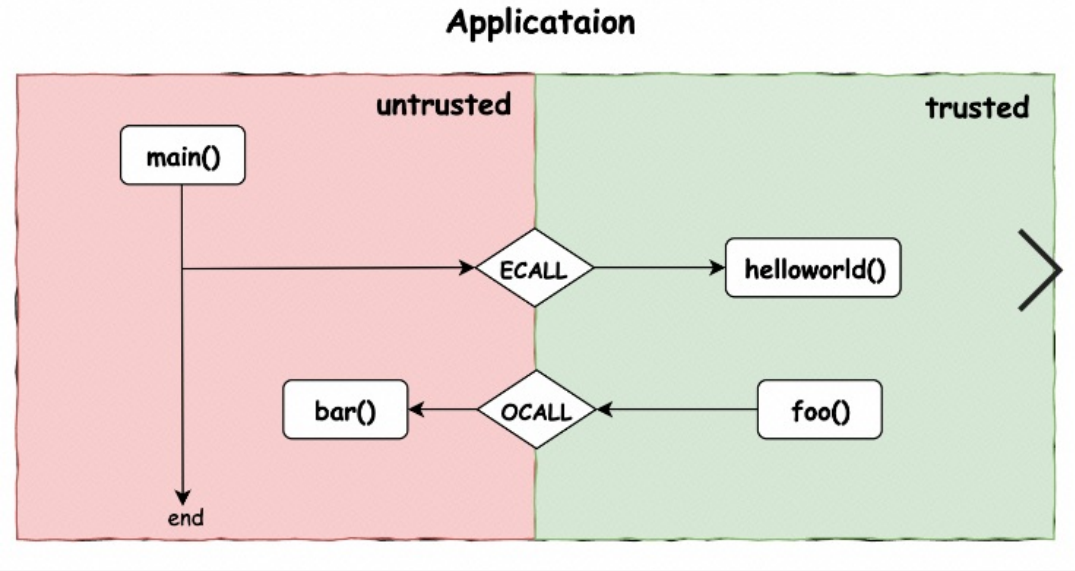
This topic describes how to use trusted execution environment (TEE) SDK to develop, build, and deploy applications. In this topic, an application named `helloworld` is used as an example. This application generates messages in an enclave on a regular basis and sends the messages to an untrusted buffer. Then, it sends the messages to terminals.

Prerequisites

- Create a managed Kubernetes cluster for confidential computing.
- The following environments are available for developing and compiling applications:
  - Aliyun Linux 2, CentOS, and RHEL7+.
  - Intel SGX Driver.
  - TEE SDK: a development kit provided by Alibaba Cloud to develop applications for confidential computing. This kit includes development models and programming interfaces that are consistent with those of Intel Linux SGX SDK. TEE SDK is compatible with Intel SGX SDK and Intel SGX Platform Software (PSW).
  - Architectural Enclave Service Manager (AESM).

How Intel SGX works

How Intel SGX works



This section describes how Intel SGX works.

- An application of Intel SGX 2.0 consists of two components:
  - Untrusted component
    The untrusted component is an unencrypted part of the memory. If you store the code and data of an application in this part, the `main()` function of the application must also be placed in the untrusted component. In the preceding figure, the `main()` and `bar()` functions are placed in the untrusted component.

  - Trusted component (enclave)
    The trusted component or enclave is an encrypted part of the memory. This component is created by the CPU, and the data and code in this component can be accessed by only the CPU. In the preceding figure, the `helloworld()` and `foo()` functions are placed in the enclave.

- To call a function in the enclave from the untrusted component, the application must perform an enclave call (ECALL).
- To call a function in the untrusted component from the enclave, the application must perform an outside call (OCALL).
- ECALLs and OCALLs are declared in Enclave Definition Language (EDL) files.

Example

In this example, an application of Intel SGX 2.0 named `helloworld` is deployed. For more information about the source code, visit GitHub. The source code includes the code for application compilation, image building, and application deployment.

Directory hierarchy of the sample code

```
sgx-device-plugin/samples/hello_world/
                  ├── Dockerfile
                  ├── Makefile
                  ├── README.md
                  └── src
                    ├── App
                    │   ├── App.cpp
                    │   └── App.h
                    ├── Enclave
                    │   ├── Enclave.config.xml
                    │   ├── Enclave.cpp
                    │   ├── Enclave.edl
                    │   ├── Enclave.h
                    │   ├── Enclave.lds
                    │   └── Enclave_private.pem
                    ├── Include
                    └── Makefile
```

Note The following table describes the `src` directory and related files.

- The App directory contains untrusted code, such as the main() function (the entry function) and code of OCALL functions.
- The Enclave directory contains trusted code, such as the code of ECALL functions.

File	Description
Enclave.edl	The EDL file.
Enclave.lds	The enclave linker script.
Enclave_private.pem	The private key that is used to sign the enclave.so file.
Enclave.config.xml	The enclave configuration file that specifies parameters, such as the stack size and whether to enable debugging.
Enclave.h and Enclave.cpp	The code that implements the trusted component.

- The Include directory contains a header file that is shared by untrusted code and trusted code.

1. In a command-line interface (CLI), connect to the managed Kubernetes cluster for confidential computing that you create and compile hello_world.

   Run the following command in a directory of the project:

   ```
   export SGX_SDK=/opt/alibaba/teesdk/intel/sgxsdk make
   ```

   A binary file named *hello_world* is generated in the root directory of the project.

```
GEN => App/Enclave_u.h
CC  <= App/Enclave_u.c
CXX <= App/App.cpp
LINK => app
GEN => Enclave/Enclave_t.h
CC  <= Enclave/Enclave_t.c
CXX <= Enclave/Enclave.cpp
LINK => enclave.so
<EnclaveConfiguration>
  <ProdID>0</ProdID>
  <ISVSVN>0</ISVSVN>
  <StackMaxSize>0x40000</StackMaxSize>
  <HeapMaxSize>0x100000</HeapMaxSize>
  <TCSNum>10</TCSNum>
  <TCSPolicy>1</TCSPolicy>
  <!-- Recommend changing 'DisableDebug' to 1 to make the enclave undebuggable for enclave release -->
  <DisableDebug>0</DisableDebug>
  <MiscSelect>0</MiscSelect>
  <MiscMask>0xFFFFFFFF</MiscMask>
</EnclaveConfiguration>
tcs_num 10, tcs_max_num 10, tcs_min_pool 1
The required memory is 3960832B.
The required memory is 0x3c7000, 3868 KB.
Succeed.
SIGN => enclave.signed.so
The project has been built in debug hardware mode.
```

Run the `./hello_world` command to start the *hello_world* application.

```
./hello_world
```

The following output is returned:

```
Wed May  6 06:53:33 2020
Hello world From SGX Enclave!
Wed May  6 06:53:34 2020
Hello world From SGX Enclave!
...
```

The following content describes how to compile the application and shows the directory hierarchy of the compiled code:

- Use a makefile to compile the application.
  a. Use the **sgx_edger8r** tool and the `sgx_ecall` function to generate untrusted code (`Enclave_u.c` and `Enclave_u.h`) in the *App* directory.
  b. Compile untrusted binary files in the *App* directory.
  c. Use the **sgx_edger8r** tool to generate trusted code (`Enclave_t.c` and `Enclave_t.h`) in the *Enclave* directory.
  d. Compile the enclave.so file. It is a trusted dynamic-link library.
  e. Use the **sgx_sign** tool to sign the trusted dynamic-link library. The name of the trusted dynamic-link library changes to enclave.signed.so.
  f. The application is compiled.
- The directory hierarchy of the compiled code.

```
sgx-device-plugin/samples/hello_world/src/
                        ├── hello_world    #[generated]
                        ├── App
                        │   ├── App.cpp
                        │   ├── App.h
                        │   ├── App.o      #[generated]
                        │   ├── Enclave_u.c #[generated]
                        │   ├── Enclave_u.h #[generated]
                        │   └── Enclave_u.o #[generated]
                        ├── Enclave
                        │   ├── Enclave.config.xml
                        │   ├── Enclave.cpp
                        │   ├── Enclave.edl
                        │   ├── Enclave.h
                        │   ├── Enclave.lds
                        │   ├── Enclave.o    #[generated]
                        │   ├── Enclave_private.pem
                        │   ├── Enclave_t.c  #[generated]
                        │   ├── Enclave_t.h  #[generated]
                        │   └── Enclave_t.o  #[generated]
                        ├── enclave.signed.so #[generated]
                        ├── enclave.so      #[generated]
                        ├── Include
                        └── Makefile
```

2. Build an image and deploy the `helloworld` application.

   Build an image for the compiled application based on **alibabatee/centos_sgx:7**. The image must contain the dynamic-link library that is required by the compiled application.
   The Dockerfile contains the following content:

   ```
   FROM alibabatee/centos_sgx:7
   COPY src/hello_world src/enclave.signed.so /app/
   WORKDIR /app
   ENTRYPOINT ["/app/hello_world"]
   ```

   i. Run the following commands to compile and build the image:

   ```
   cd sgx-device-plugin/samples/hello_world
   TARGET_IMAGE=registry-vpc.cn-shanghai.aliyuncs.com/${namespace}/${image_name}:${image_tag} ma
   ke image
   docker push registry-vpc.cn-shanghai.aliyuncs.com/${namespace}/${image_name}:${image_tag}
   ```

ii. Run the following command to deploy the `helloworld` application:

```
cat <<EOF | kubectl create -f -
apiVersion: apps/v1
kind: Deployment
metadata:
 name: helloworld
 namespace: default
spec:
 replicas: 2
 selector:
  matchLabels:
   app: helloworld
 template:
  metadata:
   labels:
    app: helloworld
  spec:
   containers:
   - command:
    - /app/hello_world
    image: {{TARGET_IMAGE}}
    imagePullPolicy: Always
    name: helloworld
    resources:
     limits:
      cpu: 250m
      memory: 512Mi
      alibabacloud.com/sgx_epc_MiB: 2
    volumeMounts:
    - mountPath: /var/run/aesmd/aesm.socket
     name: aesmsocket
   volumes:
   - hostPath:
     path: /var/run/aesmd/aesm.socket
     type: Socket
    name: aesmsocket
EOF
```

Description of the code file

File path	Description	Sample code

File path	Description	Sample code
Encalve/Enclave.edl	The EDL file that declares a public ECALL function. At least one of the following functions must be declared in the EDL file of an SGX application: • An ECALL function of the `public` type. • `trusted {...}` declares an ECALL function. • `untrusted {...}` declares an OCALL function. In this example, the application does not need to invoke an OCALL function. Therefore, only one ECALL function (`ecall_hello_from_enclave`) is declared. This ECALL function is used to create a buffer in the enclave and deploy the `helloworld` application. Then, the information in the buffer is copied to the untrusted component. The application invokes `printf` in the untrusted component to print the information.	`enclave { trusted { public void ecall_hello_from_enclave([out, size=len] char* buf, size_t len); }; };`
Enclave/Enclave.lds	-	`enclave.so { global: g_global_data_sim; g_global_data; enclave_entry; g_peak_heap_used; local: *; };`

File path	Description	Sample code
Enclave/Enclave.config.xml	-	<EnclaveConfiguration> <ProdID>0</ProdID> <ISVSVN>0</ISVSVN> <StackMaxSize>0x40000</StackMaxSize> <HeapMaxSize>0x100000</HeapMaxSize> <TCSNum>10</TCSNum> <TCSPolicy>1</TCSPolicy> <!-- Recommend changing 'DisableDebug' to 1 to make the enclave undebuggable for enclave release --> <DisableDebug>0</DisableDebug> <MiscSelect>0</MiscSelect> <MiscMask>0xFFFFFFFF</MiscMask> </EnclaveConfiguration>
Enclave/Enclave.h	In most cases, this header file is empty.	#ifndef _ENCLAVE_H_ #define _ENCLAVE_H_ #endif

Enclave/Enclave.cpp	-	#include "Enclave.h"
		#include "Enclave_t.h" /* print_string */
		#include <string.h>
		void ecall_hello_from_enclave(char *buf, size_t len)
		{
		 const char *hello = "Hello world";
		 size_t size = len;
		 if(strlen(hello) < len)
		 {
		 size = strlen(hello) + 1;
		 }
		 memcpy(buf, hello, size - 1);
		 buf[size-1] = '\0';
		}

Enclave/Enclave_private.pem	The private key that is used to sign the enclave.so file.	openssl genrsa -out Enclave/Enclave_private.pem -3 3072

File path	Description	Sample code

App/App.h	-	#ifndef _APP_H_
		#define _APP_H_
		#include <assert.h>
		#include <stdio.h>
		#include <stdlib.h>
		#include <stdarg.h>
		#include "sgx_error.h" /* sgx_status_t */
		#include "sgx_eid.h" /* sgx_enclave_id_t */
		#ifndef TRUE
		# define TRUE 1
		#endif
		#ifndef FALSE
		# define FALSE 0
		#endif
		# define TOKEN_FILENAME "enclave.token"
		# define ENCLAVE_FILENAME "enclave.signed.so"
		extern sgx_enclave_id_t global_eid; /* global enclave id */
		#if defined(__cplusplus)
		extern "C" {
		#endif
		#if defined(__cplusplus)
		}
		#endif
		#endif /* !_APP_H_ */

		#include <stdio.h>
		#include <string.h>
		#include <assert.h>
		#include <time.h>
		#include <ctime>
		# include <unistd.h>
		# include <pwd.h>
		# define MAX_PATH FILENAME_MAX
		#include "sgx_urts.h"
		#include "App.h"
		#include "Enclave_u.h"
		/* Global EID shared by multiple threads */
		sgx_enclave_id_t global_eid = 0;
		int initialize_enclave(void)
		{
		sgx_status_t ret = SGX_ERROR_UNEXPECTED;
		char enclavefile[256];
		getcwd(enclavefile, sizeof(enclavefile));
		strcat(enclavefile, "/enclave.signed.so");
		/* Call sgx_create_enclave to initialize an enclave instance */

File path	Description	Sample code

App/App.cpp	-	/* Debug Support: set 2nd parameter to 1 */
		ret = sgx_create_enclave(enclavefile, SGX_DEBUG_FLAG, NULL, NULL, &global_eid, NULL);
		if (ret != SGX_SUCCESS) {
		printf("Failed to create enclave, ret code: %d, enclave file: %s\n", ret, enclavefile);
		return -1;
		}
		return 0;
		}
		tm* get_time() {
		time_t rawtime;
		struct tm * timeinfo;
		time (&rawtime);
		timeinfo = localtime (&rawtime);
		return timeinfo;
		}
		/* Application entry */
		int SGX_CDECL main(int argc, char *argv[])
		{
		(void)(argc);
		(void)(argv);
		const size_t max_buf_len = 100;
		char buffer[max_buf_len] = {0};
		/* Initialize the enclave */
		if(initialize_enclave() < 0){
		printf("Enter a character before exit ...\n");
		getchar();
		return -1;
		}
		/* Enclave calls */
		while(1) {
		ecall_hello_from_enclave(global_eid, buffer, max_buf_len);
		printf("%s%s\n", asctime(get_time()), buffer);
		fflush(stdout);
		sleep(1);
		}
		/* Destroy the enclave */
		sgx_destroy_enclave(global_eid);
		printf("Info: SampleEnclave successfully returned.\n");
		printf("Enter a character before exit ...\n");
		getchar();
		return 0;
		}

References

- TEE-based confidential computing
- Create a managed Kubernetes cluster for confidential computing
- Intel® Software Guard Extensions Developer Guide

26.5. Deploy confidential containers in managed Kubernetes cluster for confidential computing

Inclavare Containers is the first open source container runtime intended for confidential computing in the industry. Inclavare Containers allows you to launch protected containers in a hardware-based Trusted Execution Environment (TEE) to prevent untrusted entities, such as cloud service providers (CSPs), from accessing sensitive data. In a managed Kubernetes cluster for confidential computing, you can deploy confidential containers based on Inclavare Containers. This topic describes how to deploy and run confidential containers based on Inclavare Containers in a managed Kubernetes cluster for confidential computing.

Prerequisites

- Create a managed Kubernetes cluster for confidential computing.

  > **Note** Only Elastic Compute Service (ECS) instances of the **c7t security-enhanced compute optimized instance family** support managed Kubernetes clusters for confidential computing. Make sure that these instance types are available in the selected zone.

- Connect to an ACK cluster by using kubectl.

Context

You can use conventional methods to deploy confidential containers in managed Kubernetes clusters for confidential computing. However, technology expertise in confidential computing is required and you must use the Intel Software Guard Extensions (SGX) SDK to develop and build images. Inclavare Containers can help you streamline the processes and provides you with easy access to confidential computing. Inclavare Containers is compatible with different confidential runtimes and provides a consistent user experience across standard and confidential containers. For more information, see inclavare-containers.

Step 1: Build a runtime environment to run a confidential container

> **Note** In Kubernetes, pods are scheduled to nodes at random. When you deploy confidential containers in a pod, if you do not use labels to specify the node to which you want to schedule the pod, you must build a runtime environment for confidential containers on all nodes.

1. Run the following command to install `rune` and `shim-rune`: For more information, see rune and shim-rune.

   > **Note**
   > - `rune` is a CLI that conforms to the Open Container Initiative (OCI) runtime specification. rune is used to create and run confidential containers. For more information, see runtime-container.
   > - `shim-rune` provides `shim` for `rune`. shim-rune also provides enclave signing and signature management in addition to the basic features of `shim`.

   ```
   yum-config-manager --add-repo https://mirrors.openanolis.org/inclavare-containers/alinux2-repo && \
    rpm --import https://mirrors.openanolis.org/inclavare-containers/alinux2-repo/RPM-GPG-KEY-rpm-sign && \
    yum install -y rune shim-rune
   ```

2. Configure the container engine Containerd.

   > **Note** In TEE-based confidential computing provided by Container Service for Kubernetes (ACK), the default OCI runtime supported by Containerd is runC. However, you must use a new container runtime runE to run confidential containers.

   i. Run the following command to configure the OCI runtime runE:

   ```
   cd /etc/containerd/ && \
    sed -i 's/\(default_runtime_name = \)"runc"/\1"rune"/' config.toml && \
    sed -i '/\[plugins."io.containerd.grpc.v1.cri".containerd.runtimes\]/a\\t[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.rune]' config.toml && \
    sed -i '/\[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.rune\]/a\\t  runtime_type = "io.containerd.rune.v2"' config.toml
   ```

   ii. Run the following command to restart Containerd:

   ```
   systemctl restart containerd.service
   ```

   iii. Run the following command to check whether Containerd is started:

   ```
   systemctl status containerd.service
   ```

   Expected output:

   ```
   containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; vendor preset: disabled)
     Active: active (running) since Tuesday 2021-06-15 19:16:19 CST; 5s ago
       Docs: https://containerd.io
    Process: 212462 ExecStartPre=/sbin/modprobe overlay (code=exited, status=0/SUCCESS)
   Main PID: 212464 (containerd)
      Tasks: 288
     Memory: 6.3G
     ...
   ```

3. Run the following command to install the Occlum software stack:

   > **Note** Occlum is an enclave runtime that is supported by Inclavare Containers. Inclavare Containers must work with an enclave runtime to run confidential containers.

   ```
   yum install -y occlum-pal occlum-rdfsbase-dkms
   ```

4. Run the following command to check whether `occlum-pal` is installed:

   ```
   ls /opt/occlum/build/lib/
   ```

   Expected output:

   ```
   libocclum-pal.so.0.21.0
   ```

5. Run the following command to check whether the `rdfsbase` driver is installed:

   ```
   lsmod | grep enable_rdfsbase
   ```

   Expected output:

   ```
   enable_rdfsbase    16384  0
   ```

Step 2: Build an image for the confidential container

> **Note**
> - Substeps 1 to 5: Build and package the trusted application Hello World by using Occlum.
> - Substep 6: Build a container image that contains the trusted application Hello World.

1. Run the following command to launch the Occlum development environment `occlum-app-builder`:

   > **Note** The image version that is specified in `docker.io/occlum/occlum` must be the same as the Occlum version that is installed.

   ```
   cat << EOF | kubectl apply -f -
   apiVersion: v1
   kind: Pod
   metadata:
    labels:
     run: occlum-app-builder
    name: occlum-app-builder
    namespace: default
   spec:
    hostNetwork: true
    containers:
    - command:
     - sleep
     - infinity
     image: docker.io/occlum/occlum:0.21.0-centos8.2
     imagePullPolicy: IfNotPresent
     securityContext:
      privileged: true
     name: occlum-app-builder
   EOF
   ```

2. Run the following command to log on to the `occlum-app-builder` container in the Occlum development environment:

   ```
   kubectl exec -it occlum-app-builder -c occlum-app-builder -- /bin/bash
   ```

3. Install Docker in the Occlum development environment `occlum-app-builder`.
   i. Install Docker. For more information, see Install Docker.
   ii. After Docker is installed, run the following command to run Docker:

   ```
   nohup dockerd -b docker0 --storage-driver=vfs &
   ```

   > **Note** By default, Systemd is not installed in the container. Therefore, you cannot manage Docker by using Systemd. To start Docker, you must run `nohup dockerd -b docker0 --storage-driver=vfs &`.

iii. Run the following command to verify that Docker runs as expected:

```
docker ps
```

Expected output:

```
CONTAINER ID    IMAGE       COMMAND       CREATED       STATUS       PORTS       NAMES
```

4. The following code block is an example of the Hello World application written in the C programming language:

```
cat << EOF > hello_world.c
#include <stdio.h>
#include <unistd.h>
void main(void)
{
  while (1) {
    printf("Hello World!\n");
    fflush(stdout);
    sleep(5);
  }
}
EOF
```

5. Build and package the trusted application Hello World.

i. Run the following command to compile the program with the Occlum toolchain:

```
occlum-gcc -o hello_world hello_world.c
```

ii. Run the following command to initialize `occlum_instance`:

```
occlum new occlum_instance
```

iii. Run the following command to generate a file system image of Occlum and Occlum SGX enclave:

```
cd occlum_instance && \
  cp ../hello_world image/bin/ && \
  openssl genrsa -aes128 -out occlum_key.pem -3 3072 && \
  occlum build --sign-key occlum_key.pem
```

iv. Run the following command to package the trusted application Hello World:

```
occlum package occlum_instance.tar.gz
```

6. Build a confidential container image that contains the trusted application Hello World.

i. Use the following Dockerfile template to create a file named *Dockerfile*:

```
cat << EOF >Dockerfile
FROM scratch
ADD occlum_instance.tar.gz /
ENTRYPOINT ["/bin/hello_world"]
EOF
```

ii. Run the following command to build and push the image:

```
docker build -f Dockerfile -t "<$TARGET_IMAGE>" .
docker login -p <$password> -u <$username>
docker push "<$TARGET_IMAGE>"
```

> **Note**
>
> - The variables *<$password>* and *<$username>* separately specify the password and username that are used to log on to `docker hub`.
> - The variable *<$TARGET_IMAGE>* specifies the image address.

Step 3: Deploy the confidential container

1. Exit from the Occlum development environment `occlum-app-builder`. Run the following command to create the following RuntimeClass objects: runC and runE. Then, you can deploy the confidential container in the managed Kubernetes cluster for confidential computing.

```
cat << EOF | kubectl apply -f -
apiVersion: node.k8s.io/v1beta1
handler: runc
kind: RuntimeClass
metadata:
 name: runc
---
apiVersion: node.k8s.io/v1beta1
handler: rune
kind: RuntimeClass
metadata:
 name: rune
EOF
```

2. Run the following command to deploy the confidential container:

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
 labels:
  run: occlum-helloworld
 name:  occlum-helloworld
 namespace: default
spec:
 restartPolicy: Always
 runtimeClassName: rune
 containers:
  - command:
    - /bin/hello_world
    env:
     - name: ENCLAVE_TYPE
       value: intelSgx
     - name: RUNE_CARRIER
       value: occlum
     - name: ENCLAVE_RUNTIME_LOGLEVEL
       value: info
     - name: ENCLAVE_RUNTIME_PATH
       value: /opt/occlum/build/lib/libocclum-pal.so.0.21.0
     - name: ENCLAVE_RUNTIME_ARGS
       value: occlum_instance
    image: <$TARGET_IMAGE>
    imagePullPolicy: IfNotPresent
    name: hello-world-client
 dnsPolicy: ClusterFirst
EOF
```

The following table describes the parameters in the YAML file.

Parameter	Description
ENCLAVE_TYPE	Specifies the type of hardware used for confidential computing.
RUNE_CARRIER	`shim-rune` creates and runs Occlum applications.
ENCLAVE_RUNTIME_LOGLEVEL	Specifies the log level of the runtime. Valid values: trace, debug, info, warning, error, fatal, panic, and off.
ENCLAVE_RUNTIME_PATH	Specifies the path to start the enclave runtime.
ENCLAVE_RUNTIME_ARGS	Specifies the parameters to start the enclave runtime.
TARGET_IMAGE	The container image that is specified in Substep 6 of Step 2.

Verify that the confidential container runs as expected in the managed Kubernetes cluster for confidential computing

Run the following command to check the operational log of the confidential container:

```
kubectl logs -f occlum-helloworld
```

Expected output:

```
Hello World!
Hello World!
Hello World!
```

If `Hello World!` is printed in the log of the container every five seconds, the Inclavare Containers environment is installed and the confidential container runs as expected.

> **Note** If you have any questions about confidential containers, go to issues and leave a message.

26.6. Use confidential containers to implement remote attestation in a managed Kubernetes cluster for confidential computing

Inclavare Containers implements Enclave Attestation Architecture (EAA), which is a universal and cross-platform infrastructure that is used for remote attestation. EAA can prove that sensitive workloads are running in a genuine and hardware-based Trusted Execution Environment (TEE). This topic describes how to use confidential containers to implement remote attestation in a managed Kubernetes cluster for confidential computing.

Prerequisites

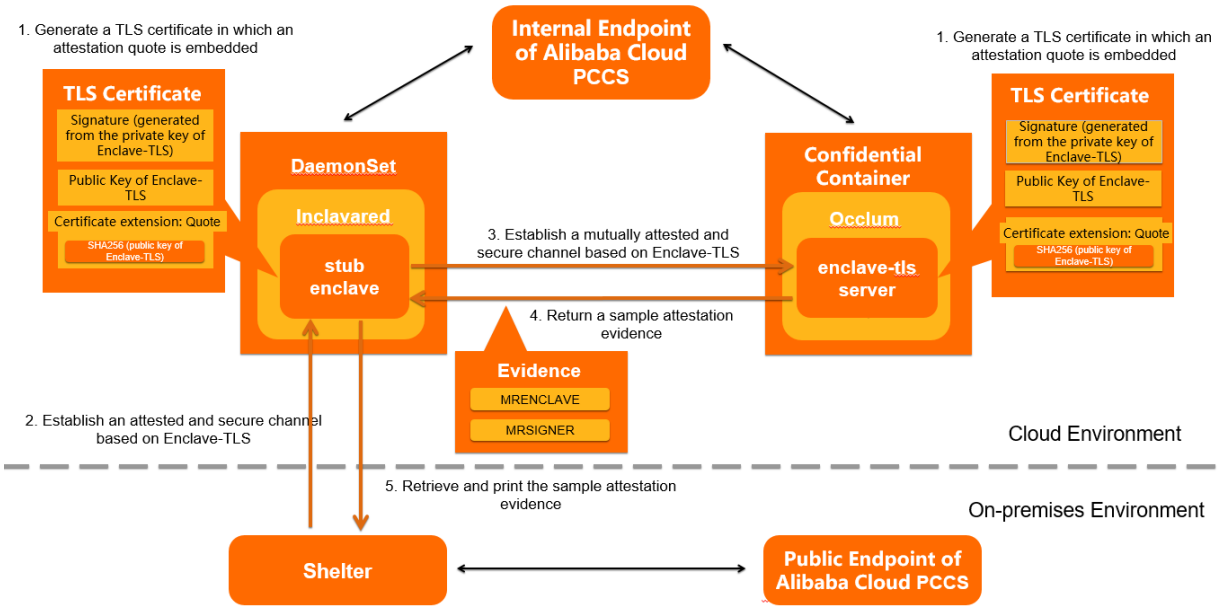
-
- An Elastic Compute Service (ECS) instance that runs CentOS 8.2 or Ubuntu 18.04 is created. For more information, see Create an ECS instance by using the provided wizard.

  > **Note** The ECS instance is used to run Shelter for remote attestation.

- Connect to an ACK cluster by using kubectl.
- A runtime environment for confidential containers is deployed in all nodes of the managed Kubernetes cluster for confidential computing. For more information, see Deploy confidential containers in managed Kubernetes cluster for confidential computing.

Context

EAA uses a TLS certificate in which a quote of a hardware-based TEE is embedded. This ensures that the server and the client communicate in a hardware-based TEE. The following figure shows the workflow and architecture of EAA.



If you want to verify that your workloads are running on a trusted platform, you can start Shelter and send the request to Inclavared for verification. The following procedure shows how to verify that your workloads are running on a trusted platform:

1. After Inclavared receives the request from Shelter, Inclavared sends the request to a confidential container. Both Inclavared and the confidential container generate a TLS certificate in which a quote of a hardware-based TEE is embedded.
2. An attested and secure channel is established between Inclavared and Shelter based on Enclave-TLS.
3. A mutually attested and secure channel is established between Inclavared and the confidential container based on Enclave-TLS.
4. Inclavared forwards the information about the hardware-based TEE and the sensitive information from the confidential container to Shelter.
5. Shelter verifies the measurements of the enclave runtime and returns the result.

The following table describes the components of EAA.

Component	Role	Description
Confidential container	Workload	Runs the server program enclave-tls-server in Occlum. The confidential container also responds to the request from Inclavared based on Enclave-TLS and returns the attestation evidence of the confidential container. The attestation evidence contains the values of `mrenclave` and `mrsigner`. For more information, see Occlum and enclave-tls.
Inclavared	Attester	Forwards the traffic between the downstream confidential container and the upstream Shelter client. The communication is protected by the attested Enclave-TLS channel.

Component	Role	Description
Shelter	On-premises verifier	1. Records the launch measurements of the enclave runtime. 2. Establishes the attested Enclave-TLS channel to communicate with Inclavared. 3. Shelter verifies the launch measurements of the enclave runtime. This way, you can verify that your workloads are running in a genuine hardware-based TEE.
Alibaba Cloud Provisioning Certificate Caching Service (PCCS)	Remote attestation service	The Alibaba Cloud SGX remote attestation service is fully compatible with the remote attestation service for Intel® SGX Elliptic Curve Digital Signature Algorithm (ECDSA) and the Intel® SGX SDK. Therefore, the vSGX instances of Alibaba Cloud, which are instances that consist of the g7t, c7t, and r7t instance types, can gain trust from remote providers and producers by using remote attestation. For more information, see Attestation Services for Intel® SGX ECDSA and the Intel® SGX SDK.

Limits

- Remote attestation can be implemented only on applications that are created by using DaemonSets. This means that an application must have one workload on each node in the cluster.
- To enable direct communication between Shelter and the workload, each node in the cluster must be associated with an elastic IP address (EIP).

Step 1: Deploy Inclavared

1. Run the following command to create an application named Inclavared by using the Daemonset:

```
cat <<-EOF | kubectl apply -f -
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: inclavared
spec:
 selector:
  matchLabels:
   k8s-app: inclavared
 template:
  metadata:
   labels:
    k8s-app: inclavared
  spec:
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: alibabacloud.com/tee-hardware-category
          operator: In
          values:
          - intel-sgx1
          - intel-sgx2
   containers:
   - image: docker.io/inclavarecontainers/inclavared:latest
    imagePullPolicy: IfNotPresent
```

```
      securityContext:
        privileged: true
      name: inclavared
      command:
        - /usr/local/bin/inclavared
      args:
        - --listen
        - 0.0.0.0:1236
        - --xfer
        - 127.0.0.1:1234
        - --attester
        - sgx_ecdsa
        - --verifier
        - sgx_ecdsa_qve
        - --mutual
      volumeMounts:
      - mountPath: /dev/sgx/enclave
        name: dev-enclave
      - mountPath: /dev/sgx/provision
        name: dev-provision
      - mountPath: /var/run/aesmd
        name: run-dir
      resources:
        requests:
          cpu: 100m
          memory: 100Mi
        limits:
          cpu: 1
          memory: 1000Mi
    tolerations:
    - effect: NoSchedule
      key: alibabacloud.com/sgx_epc_MiB
      operator: Exists
    volumes:
    - hostPath:
        path: /var/run/aesmd
        type: DirectoryOrCreate
      name: run-dir
    - hostPath:
        path: /dev/sgx_enclave
      name: dev-enclave
    - hostPath:
        path: /dev/sgx_provision
      name: dev-provision
    hostNetwork: true
EOF
```

2. Run the following command to check whether the Inclavared application is deployed:

```
kubectl get daemonset inclavared
```

Expected output:

```
NAME     DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
inclavared  2     2    2    2      2     <none>     7d22h
```

> **Note** After the Inclavared application is started:
>
> i. The listen address is set to `0.0.0.0:1236` for the Inclavared application to listen for requests from Shelter.
>
> ii. After the Inclavared application receives a request from Shelter, the Inclavared application sends the request to the workload on the current node.

Step 2: Deploy the occlum-attestation-app workload

1. Run the following command to create the workload named `occlum-attestation-app`:

```
cat <<-EOF | kubectl apply -f -
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: occlum-attestation-app
spec:
 selector:
  matchLabels:
   k8s-app: occlum-attestation-app
 template:
  metadata:
   labels:
    k8s-app: occlum-attestation-app
  spec:
   affinity:
    nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
       - key: alibabacloud.com/tee-hardware-category
         operator: In
         values:
         - intel-sgx1
         - intel-sgx2
   containers:
   - image: docker.io/inclavarecontainers/occlum-ecdsa-server:0.21.0
    imagePullPolicy: IfNotPresent
    securityContext:
     privileged: true
    name: occlum-attestation-app
    command:
     - /bin/enclave-tls-server
    args:
     - --ip
     - "127.0.0.1"   #The IP address of your on-premises machine.
     - --port
     - "1234"
     - --mutual
    env:
     - name: ENCLAVE_TYPE
      value: intelSgx
     - name: RUNE_CARRIER
      value: occlum
     - name: ENCLAVE_RUNTIME_LOGLEVEL
      value: info
```

```
        value: info
      - name: ENCLAVE_RUNTIME_PATH
        value: /opt/occlum/build/lib/libocclum-pal.so.0.21.0
      - name: ENCLAVE_RUNTIME_ARGS
        value: occlum_workspace_server
      - name: OCCLUM_RELEASE_ENCLAVE
        value: "1"
    workingDir: /run/rune
    resources:
      requests:
        cpu: 100m
        memory: 100Mi
      limits:
        cpu: 1
        memory: 1000Mi
  tolerations:
  - effect: NoSchedule
    key: alibabacloud.com/sgx_epc_MiB
    operator: Exists
  hostNetwork: true
EOF
```

2. Run the following command to check whether the `occlum-attestation-app` workload is deployed:

   ```
   kubectl get daemonset occlum-attestation-app
   ```

   Expected output:

   ```
   NAME                     DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
   occlum-attestation-app   2        2        2      2           2          <none>         7d22h
   ```

   The preceding output indicates that the `occlum-attestation-app` workload is deployed. The `enclave-tls-server` program in the workload listens for requests from Inclavared at the listen address `127.0.0.1:1234`.

Step 3: Modify the security group settings of the cluster

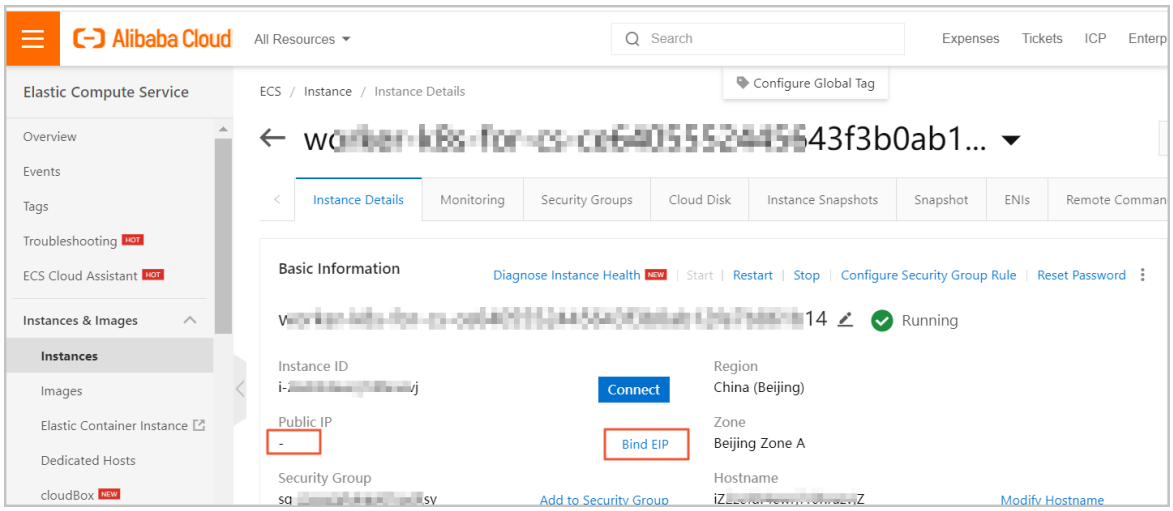
1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. On the details page of the cluster, click the **Cluster Resources** tab and click the hyperlink on the right side of Security Group.
5. Modify the default security group of the cluster and allow inbound traffic on port `1236`. For more information, see Modify security group rules.

Step 4: Associate an EIP with an ECS instance

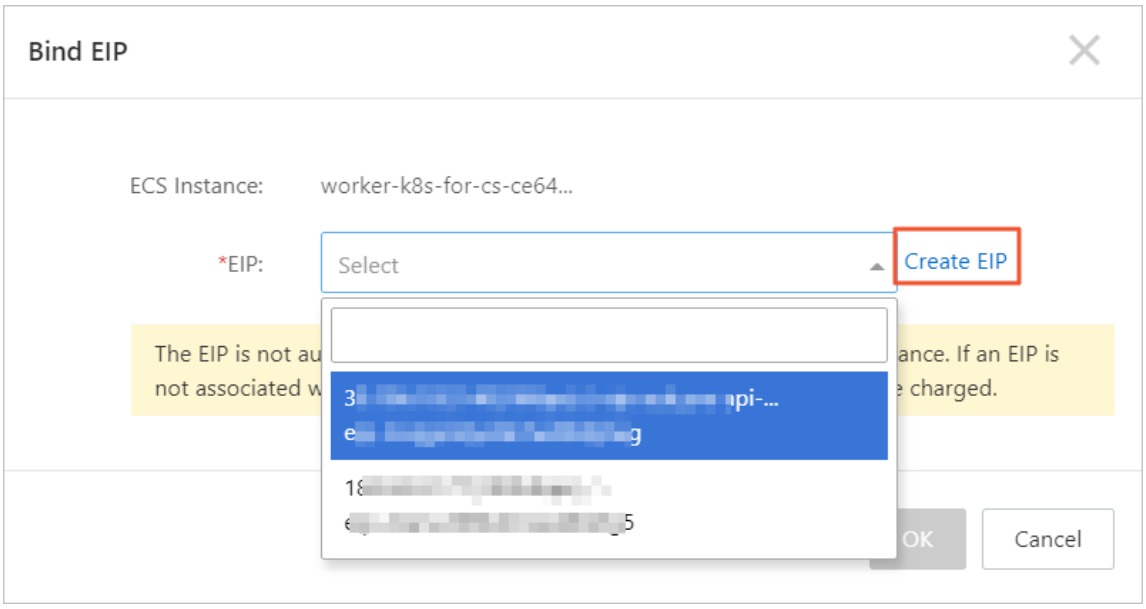
> **Note** If the ECS instance is associated with an EIP, skip this step.

1. Log on to the ACK console.
2. In the left-side navigation pane of the ACK console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.

5. On the **Nodes** page, find the node that you want to manage and click the **Instance ID** of the node.
6. In the ECS console, choose **Instances & Images > Instances**. On the page that appears, click the **Instance Details** tab.
7. Click **Bind EIP** and associate an EIP with the ECS instance.



- If you have created EIPs, select an EIP from the **EIP** drop-down list in the **Bind EIP** dialog box.
- If no EIPs are available, click **Create EIP** in the **Bind EIP** dialog box. For more information, see Apply for EIPs.



Step 5: Install Shelter for remote attestation

1. Install the SGX Platform Software (PSW).

> **Note** Shelter must rely on the dynamic library provided by the SGX PSW to verify the TLS certificate in which information about the SGX is embedded. For more information, see SGX PSW.

- On CentOS 8.2, run the following command to install the SGX PSW:

```
yum install -y yum-utils && \
  wget -c https://download.01.org/intel-sgx/sgx-linux/2.13/distro/centos8.2-server/sgx_rpm_local_repo.tg
z && \
  tar xzf sgx_rpm_local_repo.tgz && \
  yum-config-manager --add-repo sgx_rpm_local_repo && \
  yum makecache && rm -f sgx_rpm_local_repo.tgz && \
  yum install --nogpgcheck -y libsgx-dcap-quote-verify \
  libsgx-dcap-default-qpl libsgx-dcap-ql \
  libsgx-uae-service
```

- On Ubuntu 18.04, run the following command to install the SGX PSW:

```
apt-get update -y && apt-get install -y wget gnupg && \
  echo "deb [arch=amd64] https://download.01.org/intel-sgx/sgx_repo/ubuntu bionic main" | tee /etc/apt/
sources.list.d/intel-sgx.list && \
  wget -qO - https://download.01.org/intel-sgx/sgx_repo/ubuntu/intel-sgx-deb.key | apt-key add - && \
  apt-get update -y && \
  apt-get install -y libsgx-dcap-quote-verify=1.10.100.4-bionic1 \
  libsgx-dcap-default-qpl=1.10.100.4-bionic1 \
  libsgx-dcap-ql=1.10.103.1-bionic1 \
  libsgx-uae-service=2.13.100.4-bionic1
```

2. Configure the public endpoint of Alibaba Cloud PCCS.

   The Alibaba Cloud SGX remote attestation service is regionally deployed. For optimal stability, you can access this service in the region where the vSGX instance is deployed. You must manually modify the */etc/sgx_default_qcnl.conf* file to adapt to the Alibaba Cloud SGX remote attestation service that is deployed in the region where the vSGX instance is deployed.

   > **Note** Only regions in mainland China support the Alibaba Cloud SGX remote attestation service. For more information, see Regions and zones.

   - If a public IP address is assigned to the vSGX instance, modify */etc/sgx_default_qcnl.conf* to the following content:

     ```
     # PCCS server address
     PCCS_URL=https://sgx-dcap-server.<Region-ID>.aliyuncs.com/sgx/certification/v3/
     # To accept insecure HTTPS cert, set this option to FALSE
     USE_SECURE_CERT=TRUE
     ```

   - If the vSGX instance is in a virtual private cloud (VPC) and has only an internal IP address, modify */etc/sgx_default_qcnl.conf* to the following content:

     ```
     # PCCS server address
     PCCS_URL=https://sgx-dcap-server-vpc.<Region-ID>.aliyuncs.com/sgx/certification/v3/
     # To accept insecure HTTPS cert, set this option to FALSE
     USE_SECURE_CERT=TRUE
     ```

     > **Note** You must replace *<Region-ID>* with the ID of the region where the vSGX instance is deployed.

3. Install Shelter for remote attestation.

   - On CentOS 8.2, run the following command to install Shelter:

```
yum-config-manager --add-repo https://mirrors.openanolis.org/inclavare-containers/rpm-repo/ && \
  rpm --import https://mirrors.openanolis.org/inclavare-containers/rpm-repo/RPM-GPG-KEY-rpm-sign && \
  yum install -y shelter
```

- On Ubuntu 18.04, run the following command to install Shelter:

```
echo 'deb [arch=amd64] https://mirrors.openanolis.org/inclavare-containers/deb-repo bionic main' | tee /etc/apt/sources.list.d/inclavare-containers.list && \
  wget -qO - https://mirrors.openanolis.org/inclavare-containers/deb-repo/DEB-GPG-KEY.key | apt-key add - && \
  apt-get update -y && apt-get install -y shelter
```

Verify the result
Run the following command to check whether Shelter is installed:

```
which shelter
```

Expected output:

```
/usr/local/bin/shelter
```

Verify that remote attestation is implemented by using the confidential container in the managed Kubernetes cluster for confidential computing.

Run the following command on each node in the cluster to launch Shelter:

> **Note** Before you run the command, replace *<$IP>* with the EIP of the worker node.

```
shelter remoteattestation --verifier sgx_ecdsa --tls openssl --crypto openssl --addr=tcp://<$IP>:1236
```

- If Shelter is running in an SGX environment, the following output is returned:

```
root@iZ2ze[redacted]:~# shelter remoteattestation --verifier sgx_ecdsa --tls openssl --crypto openssl --addr=tcp://39.[redacted].
[redacted]

tcp sock address is [redacted]
Server:
{"id":"123456","message":"Hello and welcome to Enclave TLS!\n","mrenclave":"da375dc04c533816b8201c0a48ca9267798dd059f73f03a525067d3418b0fda2","mrsigner":"83d719e77deaca1470f6baf62a4d774303c899db69020f9c70ee1dfc08c7ce9e","msgtype":"ENCLAVEINFO","version":1}
App Enclave mrenclave: da375dc04c533816b8201c0a48ca9267798dd059f73f03a525067d3418b0fda2
App Enclave mrsigner: 83d719e77deaca1470f6baf62a4d774303c899db69020f9c70ee1dfc08c7ce9e
Remote attestation is successful.
```

  If remote attestation is implemented, `Remote attestation is successful` appears in the output.
- If Shelter is running in a non-SGX environment, the following errors are returned together with the `Remote attestation is successful` message:

```
[load_qve ../sgx_dcap_quoteverify.cpp:209] Error, call sgx_create_enclave for QvE fail [load_qve], SGXError:2006.
[sgx_qv_get_quote_supplemental_data_size ../sgx_dcap_quoteverify.cpp:527] Error, failed to load QvE.
```

  The preceding errors have no impact on the remote attestation result. You can ignore the errors.

The errors are returned because Shelter attempts to load Quote Verification Enclave (QVE) when Shelter calls sgx_qv_get_quote_supplemental_data_size(). If QVE fails to be loaded, Shelter returns the preceding errors and uses Quote Verification Library (QVL) for subsequent attestation. QVE cannot be loaded in non-SGX environments. In non-SGX environments, the attestation is handled by QVL. For more information, see Attestation services for Intel® SGX ECDSA.