Alibaba Cloud
Aliyun Container for Kubernetes

User Guide for Kubernetes Clusters

Issue: 20190801

Legal disclaimer

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information 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.
	This warning information 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 restore business.
	This indicates warning informatio n, supplementary instructions, and other content that the user must understand.	Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click **OK**.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich` *{stand | slave}*

Contents

1 Introduction

1.1 Overview

Kubernetes is a popular open-source container orchestration technology. To allow you to use Kubernetes to manage container applications in Alibaba Cloud, Alibaba Cloud Container Service provides support for Kubernetes clusters.

You can create a safe and high-availability Kubernetes cluster in the Container Service console. The Kubernetes cluster integrates with the virtualization, storage, network, and security capabilities of Alibaba Cloud to provide scalable, high-performance container application management, simplify cluster creation and expansion, and focus on the development and management of containerized applications.

Kubernetes supports the deployment, expansion, and management of containerized applications, and provides the following features:

- Elastic expansion and self-reparation.
- Service discovery and server load balancing.
- Service release and rollback.
- Secret and configuration management.

Limits

- Currently, Kubernetes clusters only support Linux containers. The support for Kubernetes Windows containers is in the works.
- Currently, Kubernetes clusters only support Virtual Private Cloud (VPC). You can select to create a VPC or use an existing VPC when creating a Kubernetes cluster.

Related open-source projects

- Alibaba Cloud Kubernetes Cloud Provider: https://github.com/AliyunContainerService/kubernetes.
- Alibaba Cloud VPC network drive for Flannel: https://github.com/coreos/flannel/blob/master/Documentation/alicloud-vpc-backend.md.

If you have any questions or suggestions regarding a specific project, you are welcome to raise an issue or pull a request in the community.

1.2 Alibaba Cloud Kubernetes vs. self-built Kubernetes

Advantages of Alibaba Cloud Kubernetes

Easy to use

- Supports creating a Kubernetes cluster with one click in the Container Service console.
- Supports upgrading Kubernetes clusters with one click in the Container Service console.

  You may have to deal with self-built Kubernetes clusters of different versions at the same time, including version 1.8.6, 1.9.4, and 1.10 in the future. Upgrading clusters each time brings you great adjustments and Operation & Maintenance (O&M) costs . Container Service upgrade solution performs rolling update by using images and uses the backup policy of complete metadata, which allows you to conveniently roll back to the previous version.
- Supports expanding or contracting Kubernetes clusters conveniently in the Container Service console.

  Container Service Kubernetes clusters allow you to expand or contract the capacity vertically with one click to respond to the peak of the data analysis business quickly.

Powerful

Function	Description
Network	· High-performance Virtual Private Cloud (VPC) network plug-in. · Supports network policy and flow control. Container Service provides you with continuous network integration and the best network optimization.

Function	Description
Server Load Balancer	Supports creating Internet or intranet Server Load Balancer instances. If your self-built Kubernetes clusters are implemented by using the self-built Ingress, releasing the business frequently may cause pressure on Ingress configuration and higher error probabilities. The Server Load Balancer solution of Container Service supports Alibaba Cloud native high-availability Server Load Balancer, and can automatically modify and update the network configurations. This solution has been used by a large number of users for a long time, which is more stable and reliable than self-built Kubernetes.
Storage	Container Service integrates with Alibaba Cloud cloud disk, Network Attached Storage (NAS), and block storage, and provides the standard FlexVolume drive. Self-built Kubernetes clusters cannot use the storage resources on the cloud. Alibaba Cloud Container Service provides the best seamless integration.
O&M	· Integrates with Alibaba Cloud Log Service and CloudMonitor. · Supports auto scaling.

Function	Description
Image repository	· High availability. Supports high concurrency. · Supports speeding up the pull of images. · Supports P2P distribution. The self-built image repository may crash if you pull images from millions of clients at the same time. Enhance the reliability of the image repository by using the image repository of Container Service, which reduces the O&M burden and upgrade pressure.
Stability	· The dedicated team guarantees the stability of the container. · Each Linux version and Kubernetes version are provided to you after strict tests. Container Service provides the Docker CE to reveal all the details and promotes the repair capabilities of Docker. If you have issues such as Docker Engine hang, network problems, and kernel compatibility, Container Service provides you with the best practices.
High availability	· Supports multiple zones. · Supports backup and disaster recovery.
Technical support	· Provides the Kubernetes upgrade capabilities. Supports upgrading a Kubernetes cluster to the latest version with one click. · Alibaba Cloud container team is responsible for solving problems about containers in your environment.

Costs and risks of self-built Kubernetes

- Building clusters is complicated

  You must manually configure the components, configuration files, certificates, keys, plug-ins, and tools related to Kubernetes. It takes several days or weeks for professional personnel to build the cluster.
- For public cloud, it takes you significant costs to integrate with cloud products.

  You must devote your own money to integrate with other products of Alibaba Cloud, such as Log Service, monitoring service, and storage management.
- The container is a systematic project, involving network, storage, operating system , orchestration, and other technologies, which requires the devotion of profession al personnel.
- The container technology is continuously developing with fast version iteration, which requires continuous upgrade and test.

2 Authorization management

2.1 Role authorization

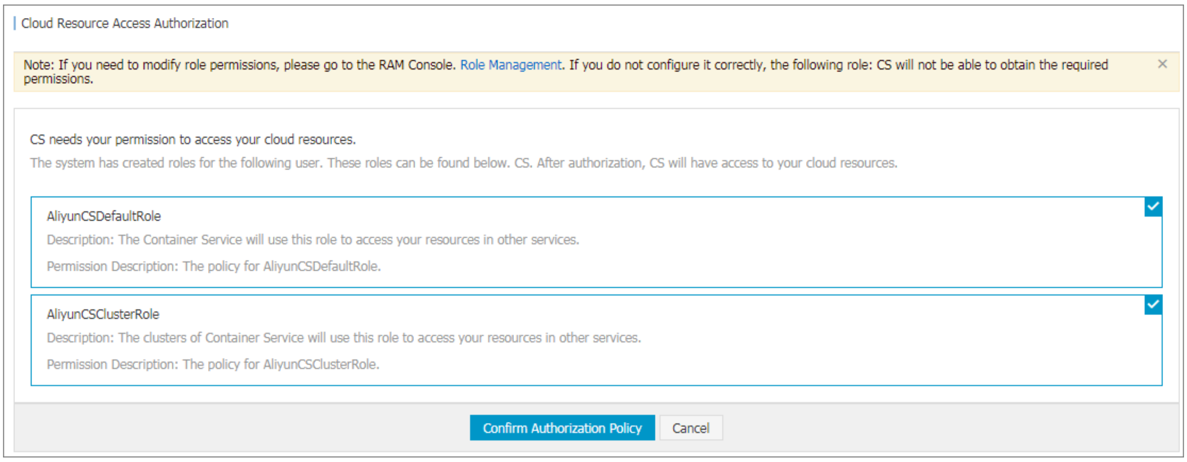
Grant the system default roles AliyunCSDefaultRole and AliyunCSClusterRole to the service account when you activate Container Service. Only after the roles are correctly granted, Container Service can normally call services such as Elastic Compute Service (ECS), Object Storage Service (OSS), Network Attached Storage (NAS), and Server Load Balancer (SLB), create clusters, and store logs.

Instructions

- If you have used Container Service before 15 January 2018, the system completes the role authorization by default. For the detailed granted permissions, see the following Default role permissions section. If you used Container Service with a Resource Access Management (RAM) user before, upgrade the authorization policy for the RAM user. For more information, see Create custom authorization policies.
- On 15 January 2018, Container Service is fully accessed to the cross-service authorization. New users who use the primary account can use Container Service only after having the cross-service authorization completed. If new users need to authorize RAM users to use Container Service, go to the RAM console to authorize the RAM users. For more information, see Use the Container Service console as a RAM user.

Procedure

1. If you have not granted the default roles to the service account correctly, the Cloud Resource Access Authorization page appears after you log on to the Container Service console. Click Confirm Authorization Policy.



Note:
Container Service has configured the default role permissions. To modify the role permissions, go to the User Management page of the RAM console. Note that incorrect configurations might cause Container Service cannot obtain the required permissions.

2. After completing the authorization, refresh the Container Service console and then perform the operations.

   To view the policy details of the roles AliyunCSDefaultRole and AliyunCSClusterRole, log on to the RAM console.

Default role permissions

For more information about permissions of each role, see the API documents of each product.

AliyunCSDefaultRole permissions

The default role AliyunCSDefaultRole contains the following main permissions:

- ECS-related permissions

Action	Description
ecs:RunInstances	Query ECS instance information.
ecs:RenewInstance	Renew ECS instances.

Action	Description
ecs:Create*	Create ECS-related resources, such as instances and disks.
ecs:AllocatePublicIpAddress	Allocate public IP addresses.
ecs:AllocateEipAddress	Allocate Elastic IP (EIP) addresses.
ecs:Delete*	Delete ECS instances.
ecs:StartInstance	Start ECS-related resources.
ecs:StopInstance	Stop ECS instances.
ecs:RebootInstance	Restart ECS instances.
ecs:Describe*	Query ECS-related resources.
ecs:AuthorizeSecurityGroup	Configure inbound security group rules.
ecs:RevokeSecurityGroup	Revoke security group rules.
ecs:AuthorizeSecurityGroupEgress	Configure outbound security group rules.
ecs:AttachDisk	Add disks.
ecs:DetachDisk	Clean up disks.
ecs:AddTags	Add tags.
ecs:ReplaceSystemDisk	Change system disks of ECS instances.
ecs:ModifyInstanceAttribute	Modify ECS instance attributes.
ecs:JoinSecurityGroup	Add ECS instances to specified security groups.
ecs:LeaveSecurityGroup	Remove ECS instances from specified security groups.
ecs:UnassociateEipAddress	Unbind EIP addresses.
ecs:ReleaseEipAddress	Release EIP addresses.

- Virtual Private Cloud (VPC)-related permissions

Permission name (Action)	Permission description
vpc:Describe*	Query information of VPC-related resources.
vpc:DescribeVpcs	Query VPC information.
vpc:AllocateEipAddress	Allocate EIP addresses.
vpc:AssociateEipAddress	Associate with EIP addresses.

Permission name (Action)	Permission description
vpc:UnassociateEipAddress	Do not associate with EIP addresses.
vpc:ReleaseEipAddress	Release EIP addresses.
vpc:CreateRouteEntry	Create router interfaces.
vpc:DeleteRouteEntry	Delete router interfaces.

- SLB-related permissions

Action	Description
slb:Describe*	Query information related to Server Load Balancer.
slb:CreateLoadBalancer	Create Server Load Balancer instances.
slb:DeleteLoadBalancer	Delete Server Load Balancer instances.
slb:RemoveBackendServers	Unbind Server Load Balancer instances.
slb:StartLoadBalancerListener	Start specified listeners.
slb:StopLoadBalancerListener	Stop specified listeners.
slb:CreateLoadBalancerTCPListener	Create TCP-based listening rules for Server Load Balancer instances.
slb:AddBackendServers	Add backend servers.

AliyunCSClusterRole permissions

The default role AliyunCSClusterRole contains the following main permissions:

- OSS-related permissions

Action	Description
oss: PutObject	Upload file or folder objects.
oss: GetObject	Get file or folder objects.
oss: ListObjects	Query file list information.

- NAS-related permissions

Action	Description
nas:Describe*	Return NAS-related information.
nas:CreateAccessRule	Create permission rules.

- SLB-related permissions

Action	Description
slb:Describe*	Query information related to Server Load Balancer.
slb:CreateLoadBalancer	Create Server Load Balancer instances.
slb:DeleteLoadBalancer	Delete Server Load Balancer instances.
slb:RemoveBackendServers	Unbind Server Load Balancer instances.
slb:StartLoadBalancerListener	Start specified listeners.
slb:StopLoadBalancerListener	Stop specified listeners.
slb:CreateLoadBalancerTCPListener	Create TCP-based listening rules for Server Load Balancer instances.
slb:AddBackendServers	Add backend servers.
slb:DeleteLoadBalancerListener	Delete listening rules of Server Load Balancer instances.
slb:CreateVServerGroup	Create VServer groups and add backend servers.
slb:ModifyVServerGroupBackendServers	Change backend servers in VServer groups.
slb:CreateLoadBalancerHTTPListener	Create HTTP-based listeners for Server Load Balancer instances.
slb:SetBackendServers	Configure backend servers and set the weight for a group of ECS instances at the Server Load Balancer instance backend.
slb:AddTags	Add tags for Server Load Balancer instances.

2.2 Use the Container Service console as a RAM user

You can log on to and perform operations in the Container Service console as a RAM user.

Before you can log on to the Container Service console and perform operations as a RAM user, you must grant related permissions to the RAM user.

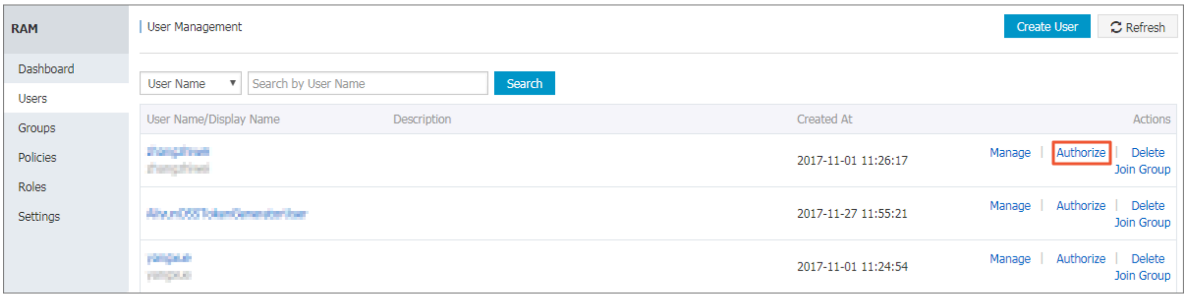
Step 1: Create a RAM user and enable console logon

1. Log on to the RAM console.

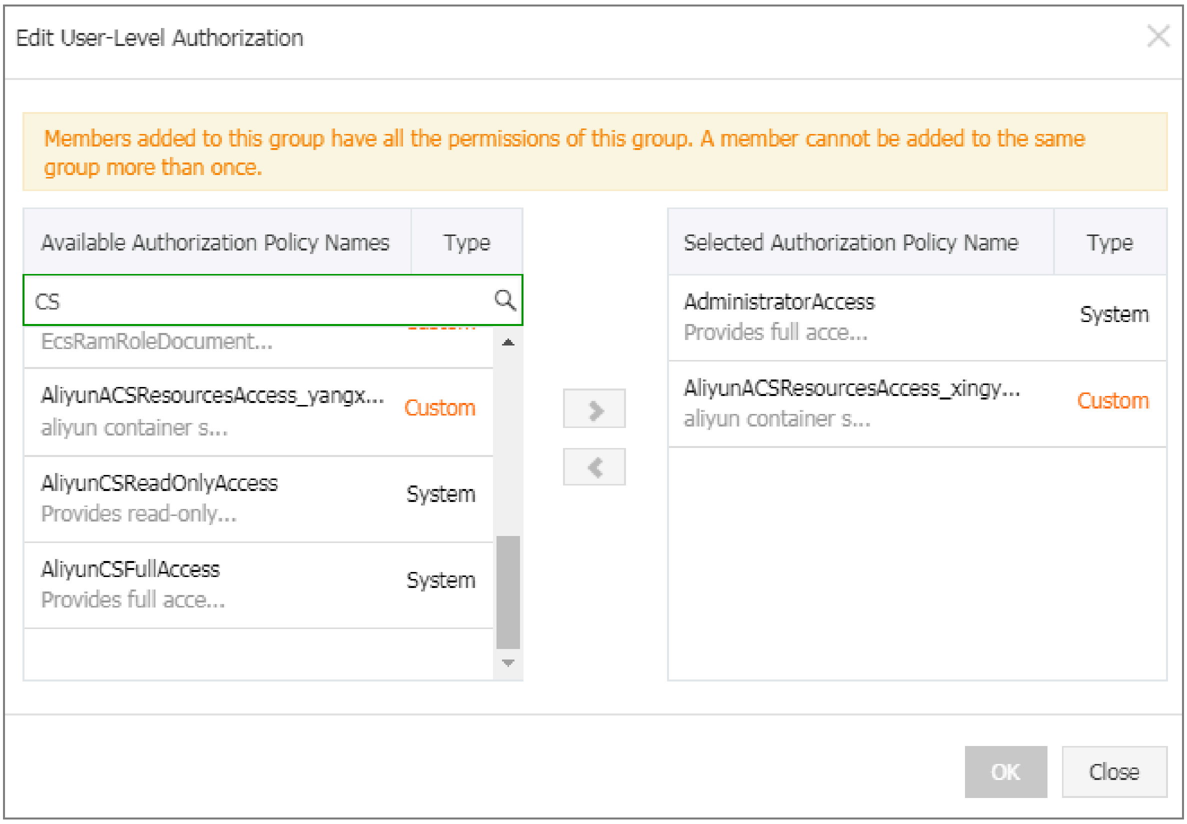
2. In the left-side navigation bar, click Users. Then, click Create User.
3. Enter a user name for the RAM user and then click OK.
4. On the Users page, select the created RAM user and click Manage.
5. In the Web Console Logon Management area, click Enable Console Logon.
6. Enter a logon password and click OK.

Step 2: Grant the RAM user permissions to access Container Service

1. On the Users page, select the created RAM user and click Authorize.



2. Select the required policies to attach them to the RAM user.



You can use the following system policies:

- AliyunCSFullAccess: Provides full access to Container Service.
- AliyunCSReadOnlyAccess: Provides read-only access to Container Service.

You can also create custom policies as you need and attach them to the RAM user. For more information, see Create custom authorization policies.

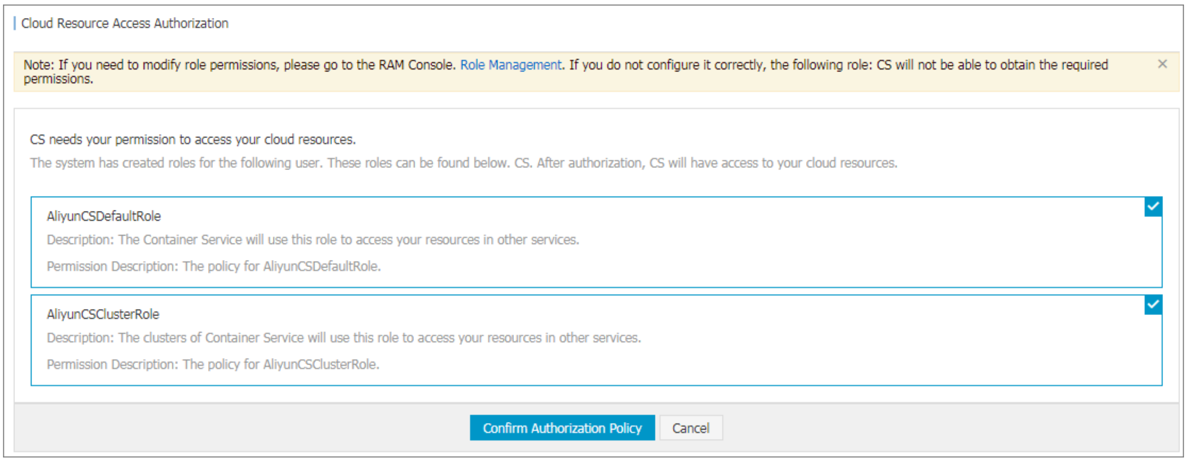
Step 3: Log on to the Container Service console as a RAM user

- If you have granted the AliyunCSDefaultRole and AliyunCSClusterRole roles to the Alibaba Cloud account, you can log on to the Container Service console and perform operations as a RAM role directly.

  Log on to the Container Service console as a RAM user.

- If you have not granted the AliyunCSDefaultRole and AliyunCSClusterRole roles to the Alibaba Cloud account, you must log on to the Container Service console using the account credentials and

  click Confirm Authorization Policy on the authorization page to grant the account the following permissions.

  After you grant the preceding permissions to the account, you can log on to the Container Service and perform related operations as a RAM user.

2.3 Grant a RAM user the permissions to access a Kubernetes cluster

This topic describes the RAM and RBAC permissions required by a RAM user to access a Kubernetes cluster, and the general steps to grant the required permissions to a RAM user.

RAM permissions

Alibaba Cloud Container Service for Kubernetes (ACK) provides the RAM permissions required to control the access to the management interface of a Kubernetes cluster. If you want to use a RAM user to scale in or scale out a Kubernetes cluster, or add nodes to the cluster, you must grant the corresponding RAM permissions to the user account. For more information, see Create custom authorization policies.

For information about the RAM permissions that can be granted to a RAM user, see #unique_8/unique_8_Connect_42_table_kla_5hy_yys.

RBAC permissions

ACK provides the RBAC permissions required by a RAM user to access the resources of a Kubernetes cluster by calling the API server of the cluster. For more information, see Authorization overview.

The following table lists the RBAC permissions that can be granted to a RAM user to access a Kubernetes cluster.

Table 2-1: RBAC permissions

Role	Permission
Admin	The read and write permissions of resources in all namespaces, and those of nodes, volumes, namespaces, and quotas.
Operation	The read and write permissions of resources in all namespaces, and the ready permissions of nodes, volumes, namespaces, and quotas.
Developer	The read and write permissions of the resources in all namespaces or specified namespaces.
Restricted User	The read permissions of resources in all namespaces or specified namespaces.
Custom	The permissions of the RAM user depend on the cluster role you select. Confirm the permissions that your selected cluster role has on resources before authorization, to avoid inappropriate permissions granted to the RAM user.

Grant a RAM user the permissions to access a Kubernetes cluster

1. Grant a RAM user the RAM permissions to access a Kubernetes cluster.

   The following are the two types of required RAM permissions:

   - Read permissions: Allow a RAM user to view basic cluster information, such as the cluster configuration and kubeconfig.
   - Write permissions: Allow a RAM user to scale in or scale out a Kubernetes cluster, upgrade the cluster, delete a node from the cluster, add a node to the cluster, and perform other actions to manage cluster resources.

   The following shows a RAM policy file that contains a read permission:

```
{
  " Statement ": [
    {
      " Action ": " cs : Get *",
      " Effect ": " Allow ",
      " Resource ": [
        " acs : cs :*:*: cluster /< yourcluste  rID >"
      ]
    }
  ],
  " Version ": " 1 "
}
```

   For information about the specific steps in the procedure, see Create custom authorization policies.

2. Grant the RBAC permissions to a RAM user to access Kubernetes resources. For information about the specific steps, see Grant RBAC permissions to a RAM user.

2.4 Create custom authorization policies

The authorization granularity of the system authorization policies provided by Container Service is coarse. If these authorization policies with coarse granularit y cannot satisfy your requirements, create the custom authorization policies. For example, to control the permissions to a specific cluster, you must use the custom authorization policy to meet the requirements with fine granularity.

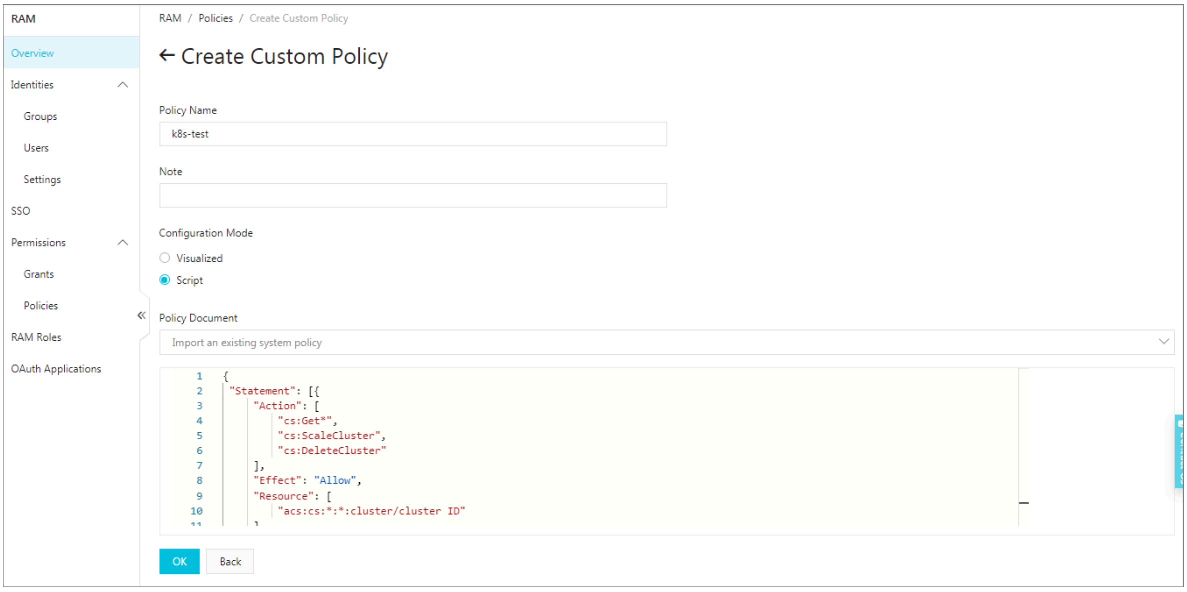
Create custom authorization policies

Get to know the basic structure and syntax of the authorization policy language before creating custom authorization policies. For more information, see ../../SP_65/DNRAM11885314/EN-US_TP_23769.dita#concept_xg5_51g_xdb.

This document introduces how to grant Resource Access Management (RAM) users permissions to query, expand, and delete clusters.

Procedure

1. Log on to the RAM console with the primary account.
2. Click Policies in the left-side navigation pane. Click Create Authorization Policy in the upper-right corner.
3. Select a template. Enter the authorization policy name and the policy content.



```
{
 " Statement ": [{
     " Action ": [
         " cs : Get *",
         " cs : ScaleClust  er ",
         " cs : DeleteClus  ter "
     ],
     " Effect ": " Allow ",
     " Resource ": [
         " acs : cs :*:*: cluster / cluster   ID "
     ]
 }],
 " Version ": " 1 "
}
```

where:

- `Action` : Enter the permission that you want to grant.

Note:

All the Actions support wildcards.

- `Resource` supports the following configuration methods.

  - Grant permissions of a single cluster

    ```
    " Resource ": [
         " acs : cs :*:*: cluster / cluster   ID "
     ]
    ```

  - Grant permissions of multiple clusters

    ```
    " Resource ": [
         " acs : cs :*:*: cluster / cluster   ID ",
         " acs : cs :*:*: cluster / cluster   ID "
     ]
    ```

  - Grant permissions of all your clusters

    ```
    " Resource ": [
         "*"
     ]
    ```

  You must replace `cluster   ID` with your actual cluster ID.

4. Click Create Authorization Policy after completing the configurations.

Table 2-2: Container Service RAM action

Action	Description
CreateCluster	Create clusters.
AttachInstances	Add existing Elastic Compute Service (ECS) instances to clusters.
ScaleCluster	Expand clusters.
GetClusters	View cluster list.
GetClusterById	View cluster details.
ModifyClusterName	Modify cluster names.
DeleteCluster	Delete clusters.
UpgradeClusterAgent	Upgrade cluster Agent.
GetClusterLogs	View cluster operation logs.
GetClusterEndpoint	View cluster access point.
GetClusterCerts	Download cluster certificate.
RevokeClusterCerts	Revoke cluster certificate.

Action	Description
BindSLB	Bind Server Load Balancer instances to clusters.
UnBindSLB	Unbind Server Load Balancer instances from clusters.
ReBindSecurityGroup	Rebind security groups to clusters.
CheckSecurityGroup	Check existing security group rules of clusters.
FixSecurityGroup	Fix cluster security group rules.
ResetClusterNode	Reset cluster nodes.
DeleteClusterNode	Delete cluster nodes.
CreateAutoScale	Create node auto scaling rules.
UpdateAutoScale	Update node auto scaling rules.
DeleteAutoScale	Delete node auto scaling rules.
GetClusterProjects	View applications in clusters.
CreateTriggerHook	Create triggers for applications.
GetTriggerHook	View application trigger list.
RevokeTriggerHook	Delete application triggers.
CreateClusterToken	Create tokens.

2.5 Grant RBAC permissions to a RAM user

This topic describes how to grant a RAM user the RBAC permissions to access a Kubernetes cluster.

Prerequisites

You can follow the steps in this guide if your existing RAM user account meets the following requirements:

- An Alibaba Cloud account is obtained, and one or more RAM users are created.
- The target RAM user is granted the read permissions to the target Kubernetes cluster with the RAM console.
- The target RAM user is granted the required RAM permissions. For more information, see Create custom authorization policies.

- The target RAM user is granted the preset admin role or the custom cluster-admin role in a cluster or namespace. For more information, see Procedure.

Note:

- If a RAM user wants to grant permissions to other RAM users in the same cluster or namespace, the RAM user must be granted the required RAM permissions. For more information, see Create custom authorization policies.
- If RAM authorization is involved in configuring permissions through the Container Service console, you must manually perform authorization in the RAM console for the target RAM user according to the reference policy and operation instructions on the page due the security restrictions of RAM.

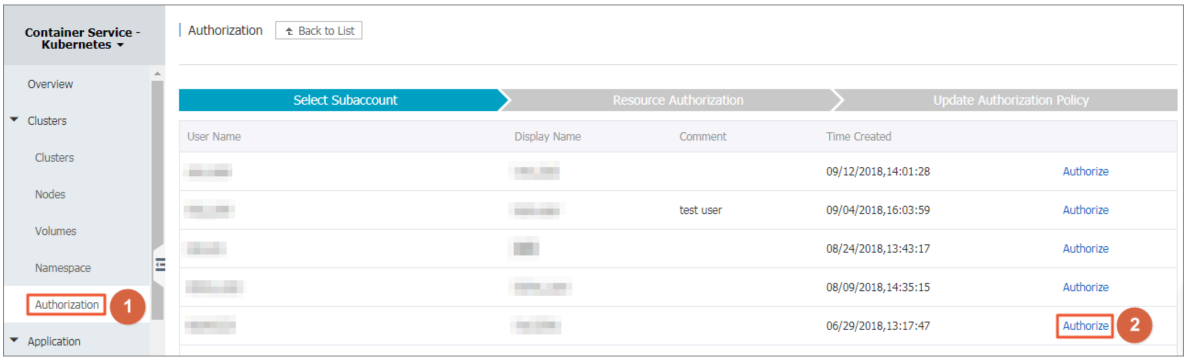
Authorization policy upgrade notice

Container Service has upgraded the cluster authorization policy to enhance the security of Kubernetes clusters. Any RAM users in a Kubernetes cluster that are not granted the required permissions cannot access the cluster resources.

Therefore, we recommend that you grant required permissions to the RAM users in each of your Kubernetes clusters. After you complete this process, your managed RAM users will only have the specified permissions to access the their corresponding authorized cluster.

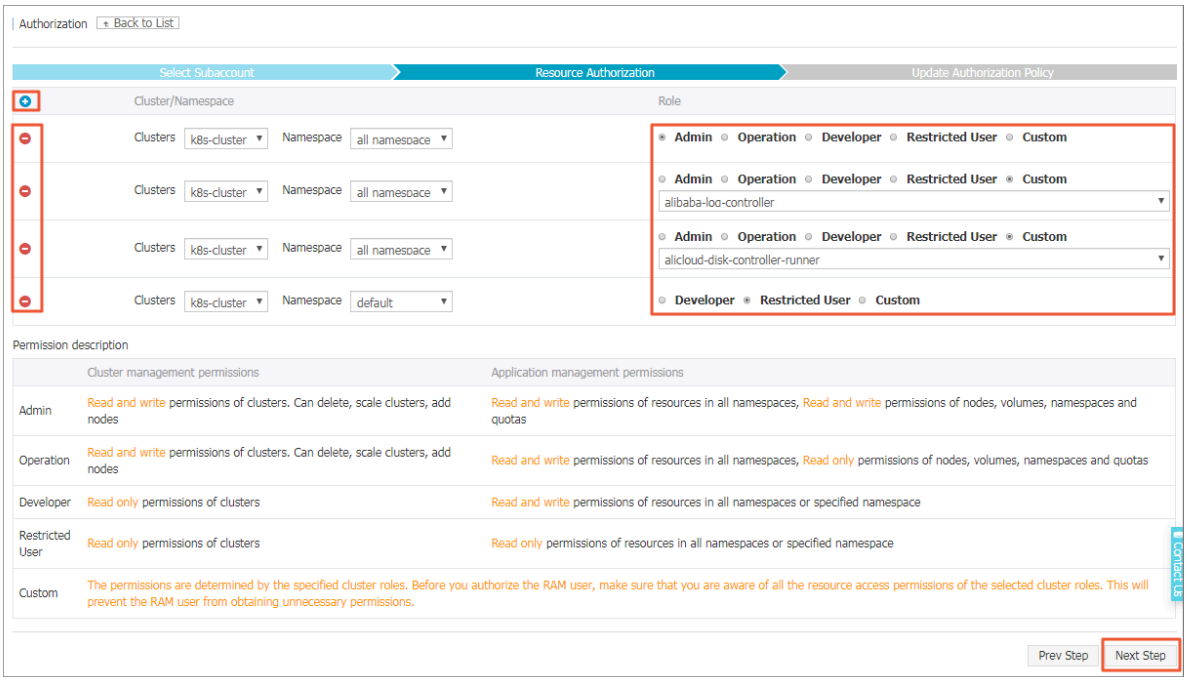
Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service - Kubernetes, choose Clusters > Authorization.
3. On the right of the target RAM user, click Authorize.



4. In the upper-left corner of the Resource Authorization tab page, click the plus sign, set the cluster and namespace where the permissions to be granted apply, set a role for the RAM user, and then click Next Step.

> **Note:**
>
> - You can grant one preset role and multiple custom roles to a RAM user in a cluster or a namespace.
> - You can click the minus sign to remove a group of permission settings (that is, the settings of the cluster, the namespace, and the role).



The following table lists the RBAC permissions that can be granted to a RAM user to access a Kubernetes cluster.

Table 2-3: RBAC permissions

Role	Permissions
Admin	The read and write permissions of resources in all namespaces, and those of nodes, volumes, namespaces, and quotas.
Operation	The read and write permissions of resources in all namespaces, and the ready permissions of nodes, volumes, namespaces, and quotas.

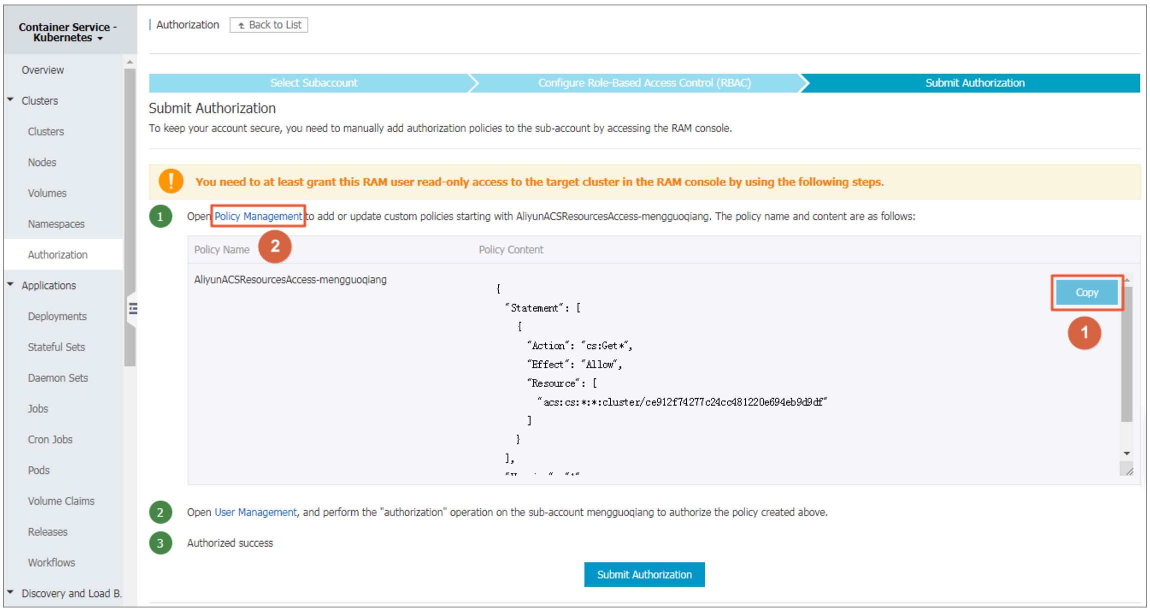
Role	Permissions
Developer	The read and write permissions of the resources in all namespaces or specified namespaces.
Restricted User	The read permissions of resources in all namespaces or specified namespaces.
Custom	The permissions of the RAM user depend on the cluster role you select. Confirm the permissions that your selected cluster role has on resources before authorization, to avoid inappropriate permissions granted to the RAM user. For more information, see Custom permissions.

5. If Authorized success is displayed, this means that you have granted the target RAM user the permissions. If the Submit Authorization page is displayed, follow these steps to use the RAM console to grant the target RAM user the read permission to a specific cluster:

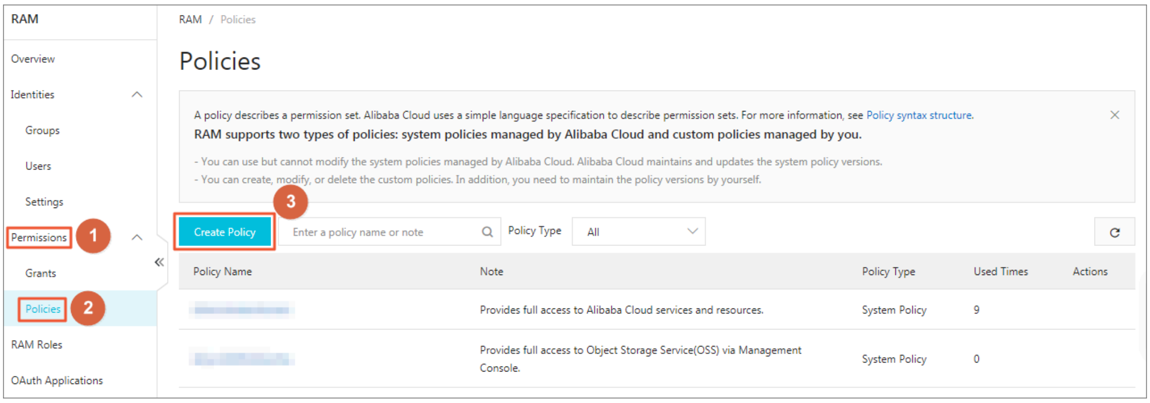
   Note:

For information about more permissions, see #unique_8/unique_8_Connect_42_table_kla_5hy_yys.

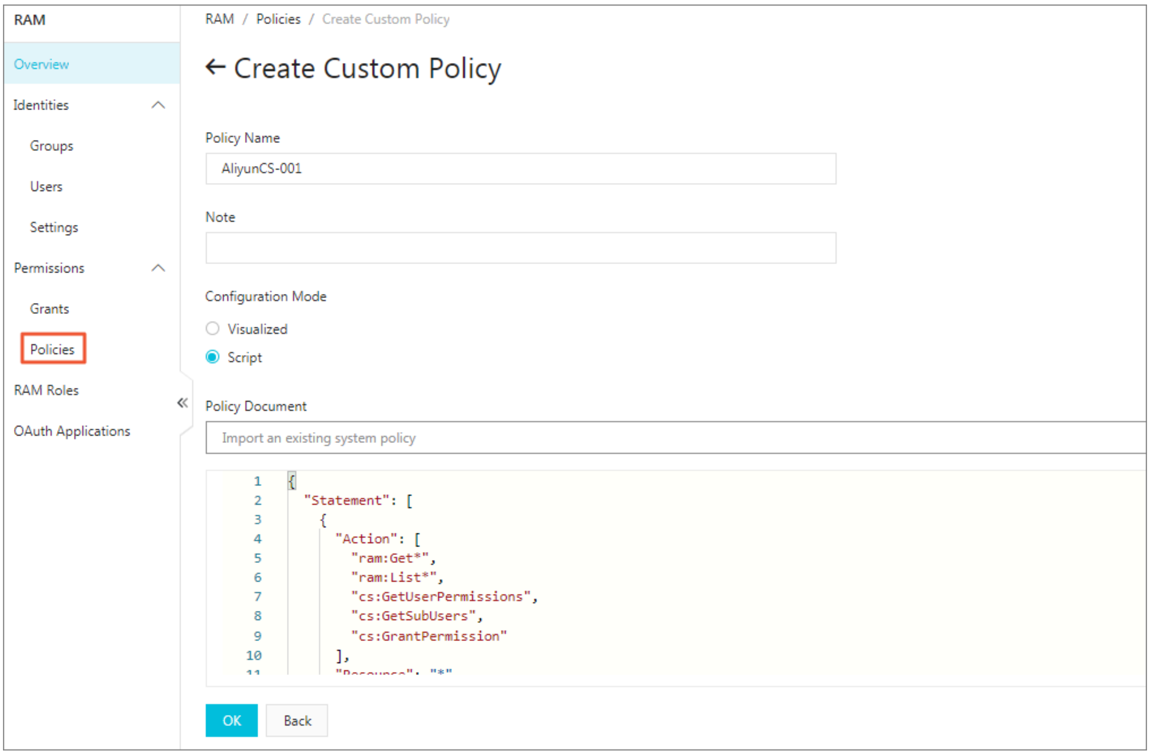
a. Click Copy, and then click Policy Management.



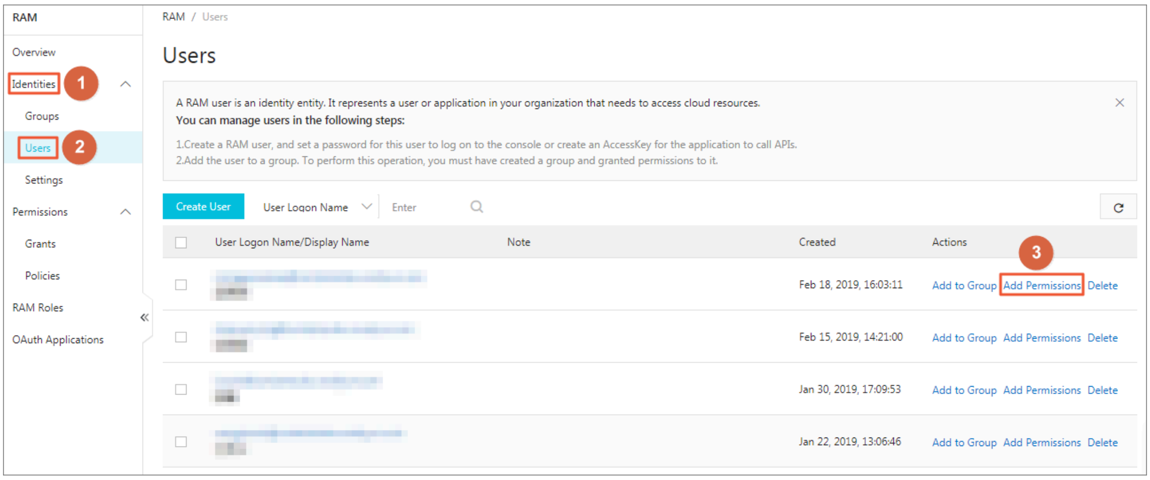
b. Choose Permissions > Policies, and then click Create Policy.



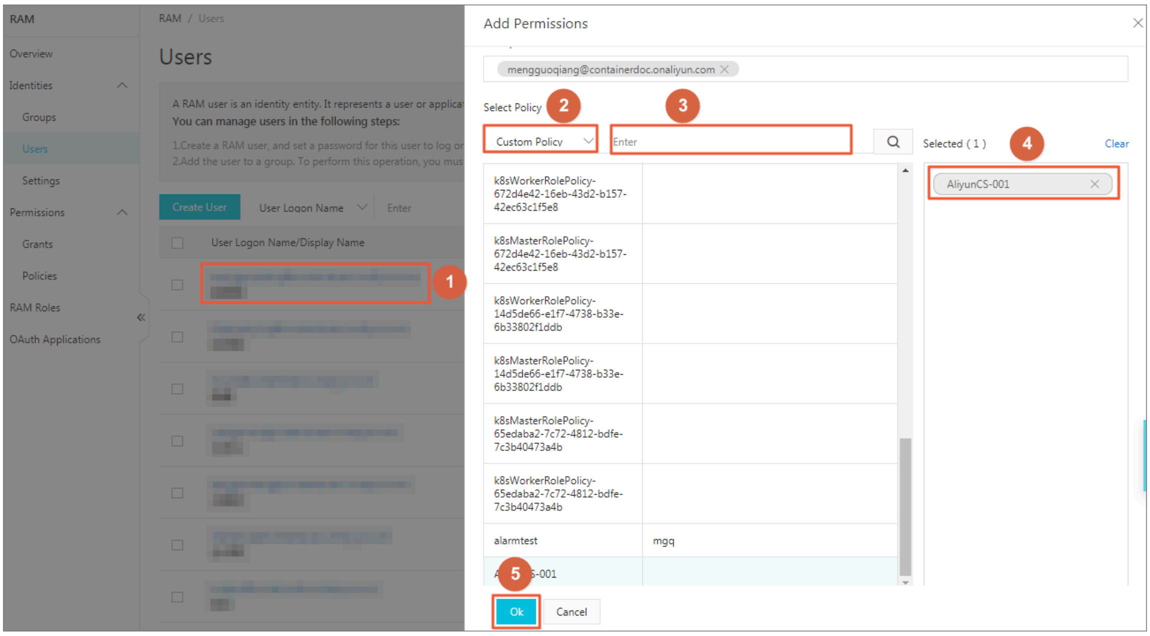
c. Enter a Policy Name, select the Script configuration mode, use the hot key Ctrl +V to paste the content that was copied in step 6 in the Policy Document area, and then click OK. For more information, see Create custom authorization policies.



d. In the left-side navigation pane, choose Identities > Users. On the right of the target user, click Add Permissions.



e. Select Custom Policy, search for or manually look for the customized policy, click the policy name to add the policy to the Selected area on the right, and then click OK.



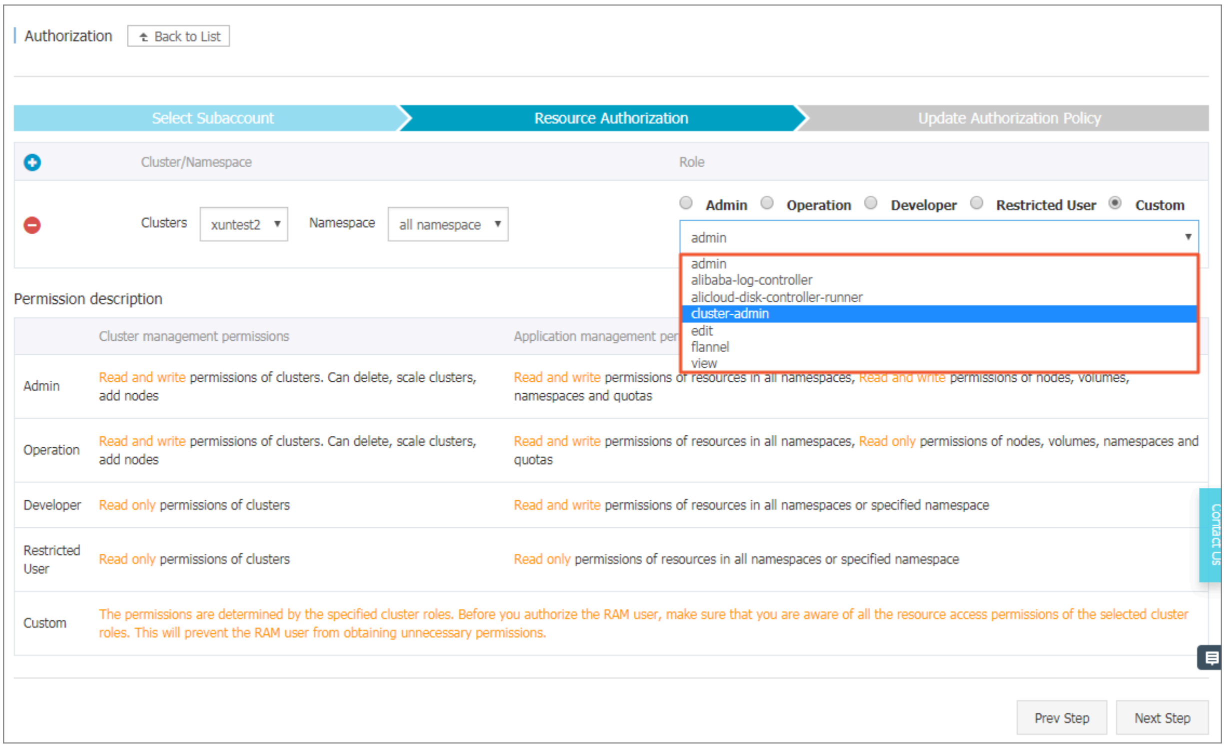
f. Return to the Submit Authorization page in the Container Service console, click Submit Authorization.

6. After you complete these steps, you can use the target RAM user to log on to the Container Service console and perform the operations allowed by the granted permissions.

Custom permissions

Alibaba Cloud Container Service offers four types of permissions by pre-setting four types of roles: Admin, Operation, Developer, and Restricted User. These types of permissions can meet the needs of most users in the Container Service console. However, if you want to customize the access permissions to clusters, you can also use the custom permissions.

> **Note:**
> Alibaba Cloud Container Service provides several custom permission. Among them, the cluster-admin permission is a super administrator permission with the permissions to access and operate on all resources.



You can log on to the cluster Master node and run the following command to view the details of the custom permissions.

```
# kubectl get clusterrol e
 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
```

To view the permission details of the super administrator cluster-admin, run the following command.

Note:
After the RAM user is granted the cluster-admin role, the RAM user can be regarded as a super administrator that has the same privileges as the Alibaba Cloud account, and it can perform operations on any resources in the cluster. Execute caution when you grant the cluster-admin role.

```
#  kubectl   get   clusterrol  e   cluster - admin   - o   yaml
 apiVersion :  rbac . authorizat  ion . k8s . io / v1
 kind :  ClusterRol  e
 metadata :
   annotation  s :
     rbac . authorizat  ion . kubernetes . io / autoupdate : " true "
   creationTi  mestamp :  2018 - 10 - 12T08 : 31 : 15Z
   labels :
     kubernetes . io / bootstrapp  ing :  rbac - defaults
   name :  cluster - admin
   resourceVe  rsion : " 57 "
   selfLink : / apis / rbac . authorizat  ion . k8s . io / v1 /
 clusterrol  es / cluster - admin
   uid :  2f29f9c5 - cdf9 - 11e8 - 84bf - 00163e0b2f  97
 rules :
-  apiGroups :
  - '*'
   resources :
  - '*'
   verbs :
  - '*'
-  nonResourc  eURLs :
  - '*'
   verbs :
  - '*'
```

3 Node management

3.1 Add an existing ECS instance to a Kubernetes cluster

You can add existing Elastic Compute Service (ECS) instances to a Kubernetes cluster. Kubernetes clusters only support adding worker nodes.

Prerequisites

- If you have not created a cluster before, create a cluster first. For how to create a cluster, see Create a Kubernetes cluster.
- Add the ECS instance to the security group of the Kubernetes cluster.

Context

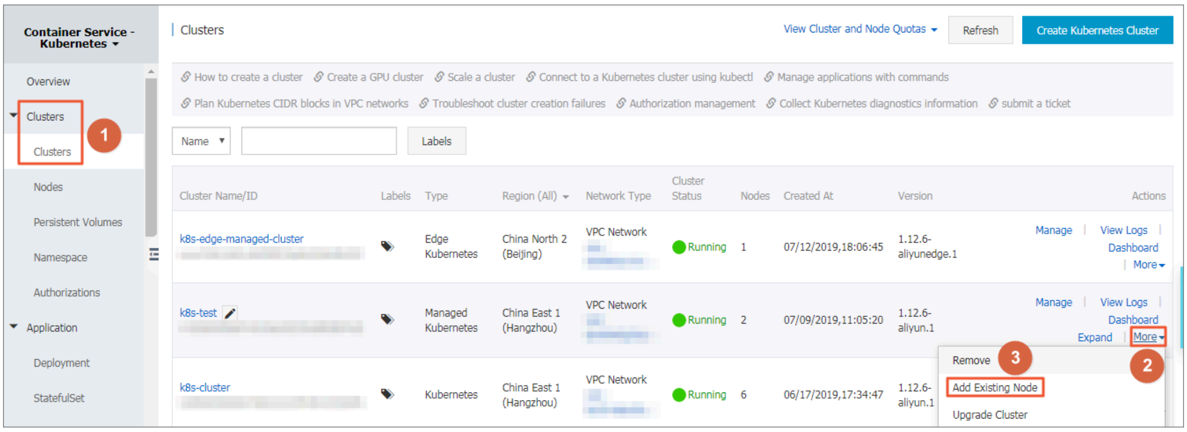
- By default, each cluster can contain up to 40 nodes. To add more nodes, open a ticket.
- The ECS instance to be added must be in the same Virtual Private Cloud (VPC) region as the cluster.
- When adding an existing instance, make sure that your instance has an Elastic IP (EIP) for the VPC network type, or the corresponding VPC is already configured with the NAT gateway. In short, make sure the corresponding node can access public network normally. Otherwise, the ECS instance fails to be added.
- The ECS instance to be added must be under the same account as the cluster.
- Only the ECS instance whose operating system is CentOS can be added.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.
3. Select the target cluster and click More > Add Existing Node.

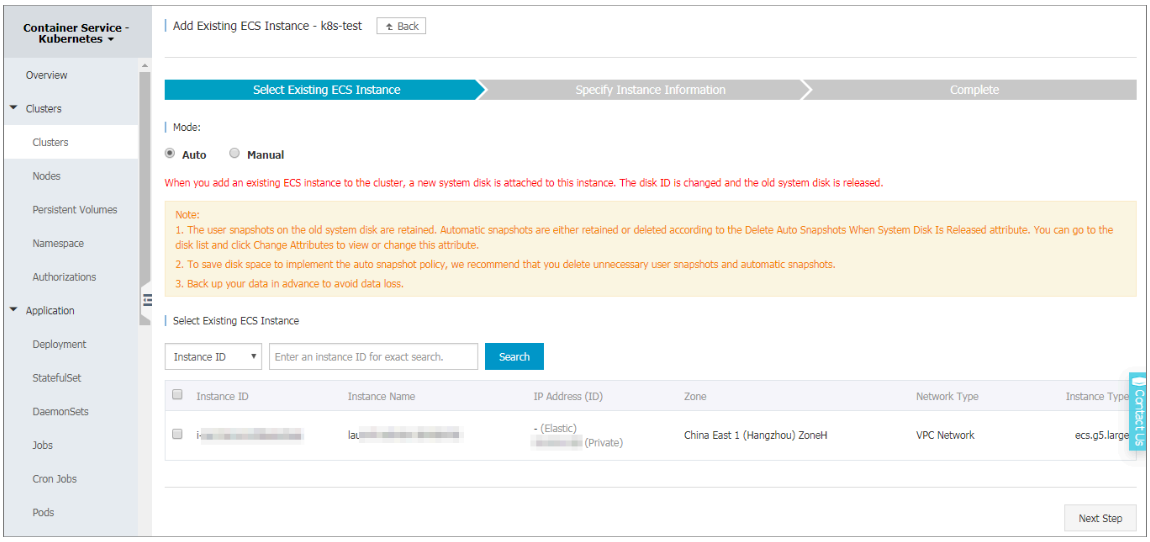
   The Add Existing ECS Instance page appears. All the available ECS instances under the current account are displayed on this page. Select to add existing ECS instances automatically or manually.

   If Automatically Add is selected, select the ECS instances to add them to the cluster automatically. If Manually Add is selected, you must obtain the command and then log on to the corresponding ECS instance to add the ECS instance to this cluster. You can only add one ECS instance at a time.

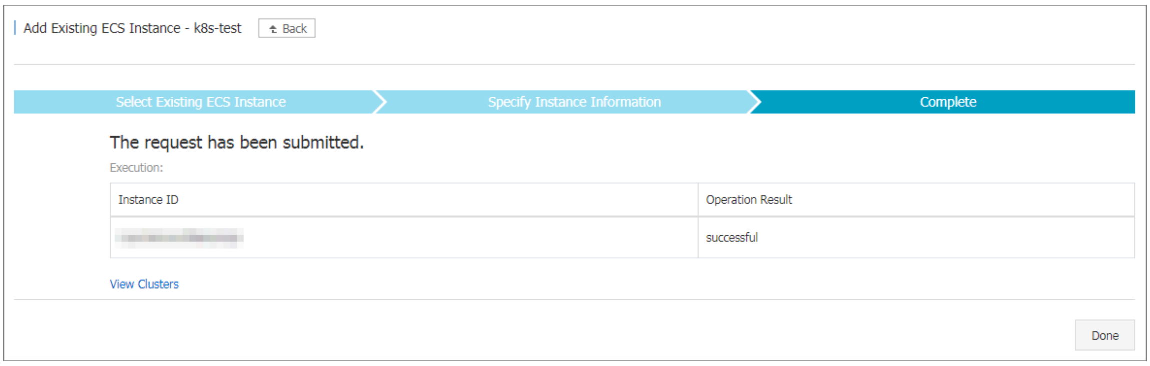


4. Select Auto to add multiple ECS instances at a time.

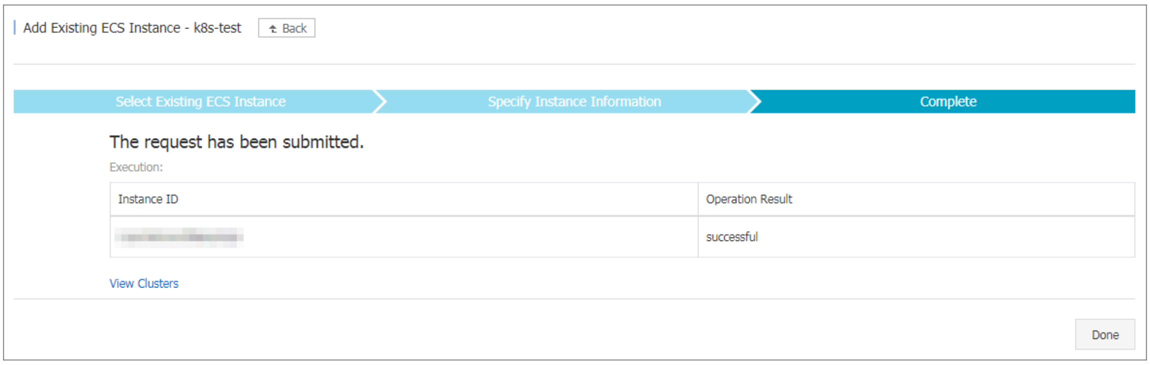
    a) In the list of existing cloud servers, select the target ECS instance, and then click Next Step.



    b) Set the password used to log on to the ECS instance, add tags to the ECS instance, and then click Next Step.

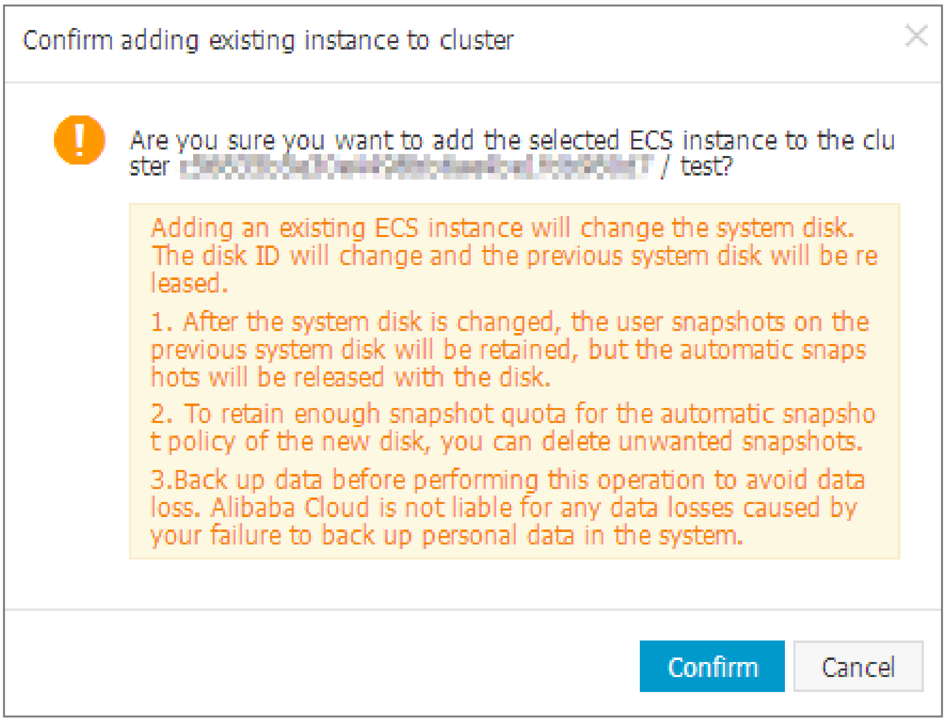


    c) Click Confirm in the displayed dialog box. The selected ECS instances are automatically added to this cluster.

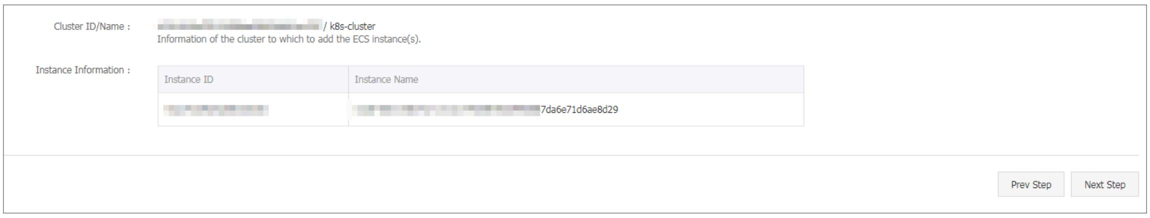


5. Optional: You can also select Manual to manually add an existing ECS instance to the cluster.

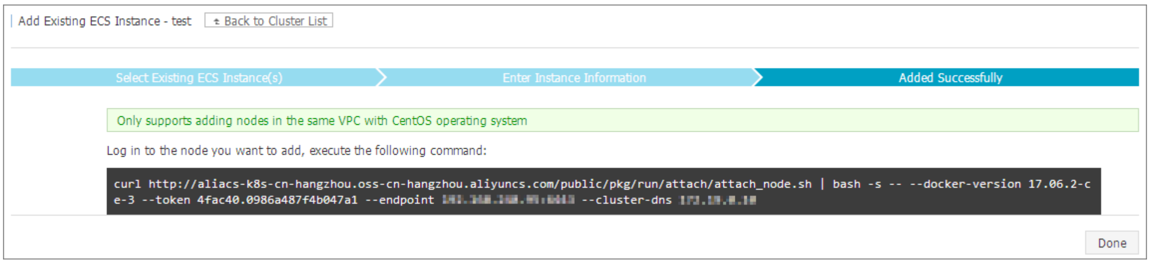
   a) Select the ECS instance to be added and then click Next Step. You can add only one ECS instance at a time.



   b) Confirm the information and then click Next Step.



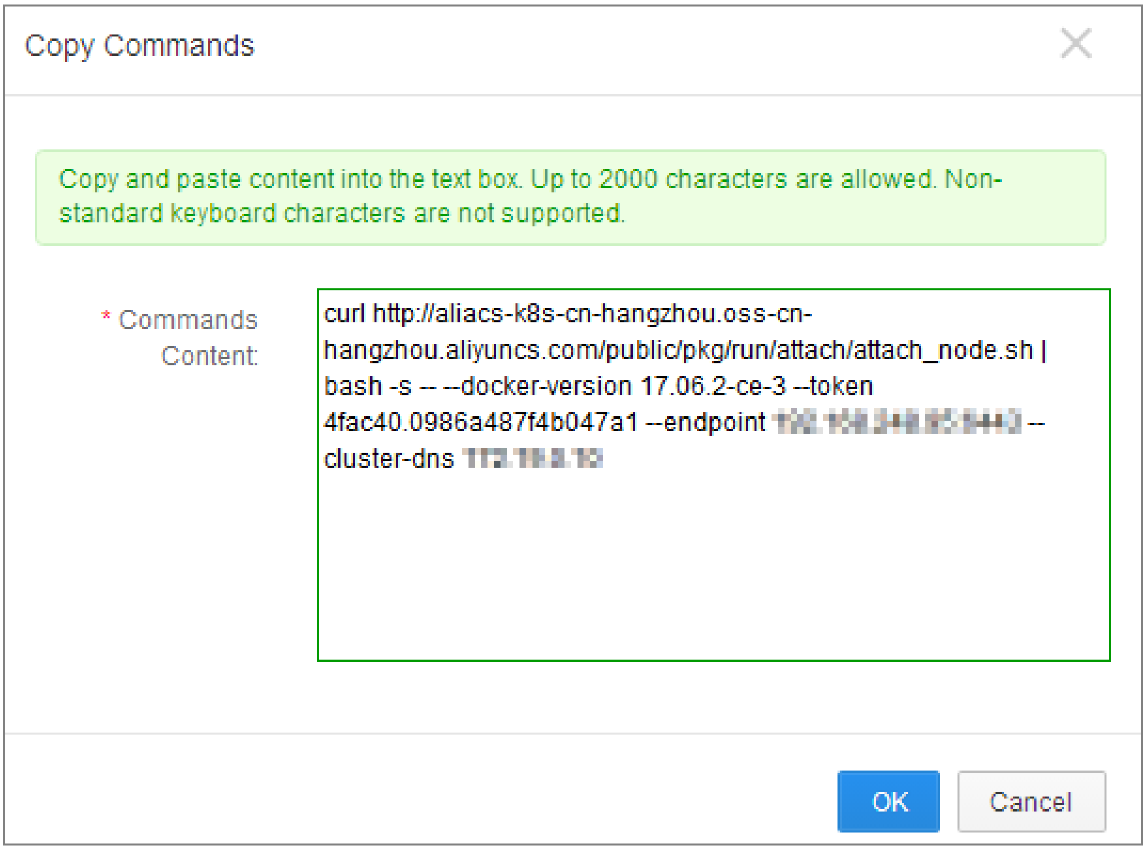
   c) Copy the command.



   d) Click Done.

   e) Log on to the ECS console and click Instances in the left-side navigation pane. Select the region in which the cluster resides and the ECS instance to be added.

f) Click Connect at the right of the ECS instance to be added. The Enter VNC Password dialog box appears. Enter the VNC password and then click OK. Enter the copied command and then click OK to run the script.



g) After the script is successfully run, the ECS instance is added to the cluster. You can click the cluster ID on the Cluster List page to view the node list of the cluster and check if the ECS instance is successfully added to the cluster.

3.2 View node list

You can view the node list of the Kubernetes cluster by using commands, in the Container Service console, or in the Kubernetes dashboard.

View node list by using commands

Note:
Before using commands to view the node list of the Kubernetes cluster, #unique_19 first.

After connecting to the Kubernetes cluster by using kubectl, run the following command to view the nodes in the cluster:

```
kubectl  get  nodes
```

Sample output:

```
$ kubectl  get  nodes
 NAME   STATUS   AGE   VERSION
 iz2ze2n6ep  53tch701yh  9zz   Ready   19m   v1 . 6 . 1 - 2 +
 ed9e3d33a0  7093
 iz2zeafr76  2wibijx39e  5az   Ready   7m   v1 . 6 . 1 - 2 +
 ed9e3d33a0  7093
 iz2zeafr76  2wibijx39e  5bz   Ready   7m   v1 . 6 . 1 - 2 +
 ed9e3d33a0  7093
 iz2zef4dnn  9nos8elyr3  2kz   Ready   14m   v1 . 6 . 1 - 2 +
 ed9e3d33a0  7093
 iz2zeitvvo  8enoreufst  kmz   Ready   11m   v1 . 6 . 1 - 2 +
 ed9e3d33a0  7093
```

View node list in Container Service console

1. Log on to the Container Service console.
2. Click Kubernetes >Clusters > > Nodesin the left-side navigation pane.
3. Select the cluster from the Cluster drop-down list and then view the node list of this cluster.

View node list in Kubernetes dashboard

1. Log on to the Container Service console.
2. Click Kubernetes > Clusters in the left-side navigation pane.
3. Click Dashboard at the right of the cluster to enter the Kubernetes dashboard.

4. In the Kubernetes dashboard, click Nodes in the left-side navigation pane to view the node list of this cluster.

3.3 Node monitoring

Kubernetes clusters integrate with the Alibaba Cloud monitoring service seamlessly. You can view the monitoring information of Kubernetes nodes and get to know the node monitoring metrics of the Elastic Compute Service (ECS) instances under Kubernetes clusters.

Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Clusters > Nodes to enter the Node List page.
3. Select the target cluster and node under the cluster.
4. Click Monitor at the right of the node to view the monitoring information of this node.
5. You are redirected to the CloudMonitor console. View the basic monitoring information of the corresponding ECS instance, including the CPU usage, network inbound bandwidth, network outbound bandwidth, disk BPS, and disk IOPS.

What's next

To view the monitoring metrics at the operating system level, install the CloudMonitor component. For more information, see Host monitoring overview.

Kubernetes clusters can now monitor resources by using application groups. For more information, see #unique_22.

3.4 Manage node labels

You can manage node labels in the Container Service console, including adding node labels in batches, filtering nodes by using a label, and deleting a node label quickly.

For how to use node labels to schedule pods to specified nodes, see #unique_24.

Prerequisite

You have successfully created a Kubernetes cluster. For more information, see #unique_25.

Add node labels in batches

1. Log on to the Container Service console.
2. Click Kubernetes Clusters > Nodes in the left-side navigation pane.
3. Select the cluster from the Clusters drop-down list and then click Label Management in the upper-right corner.

4. Select one or more nodes by selecting the corresponding check boxes and then click Add Tag.

5. Ener the name and value of the label in the displayed dialog box and then click OK.

   Nodes with the same label are displayed on the Label Management page.

Filter nodes by using a label

1. Log on to the Container Service console.
2. Click Kubernetes Clusters > Nodes in the left-side navigation pane.
3. Select the cluster from the Clusters drop-down list and then click Label Management in the upper-right corner.

4. Click the label at the right of a node to filter nodes by using the label. In this example, click `group : worker`.

   Nodes with the label `group : worker` are filtered.

Delete a node label

1. Log on to the Container Service console.
2. Click Kubernetes Clusters > Nodes in the left-side navigation pane.
3. Select the cluster from the Clusters drop-down list and then click Label Management in the upper-right corner.

4. Click the delete (x) button of a node label, for example, `group : worker`.

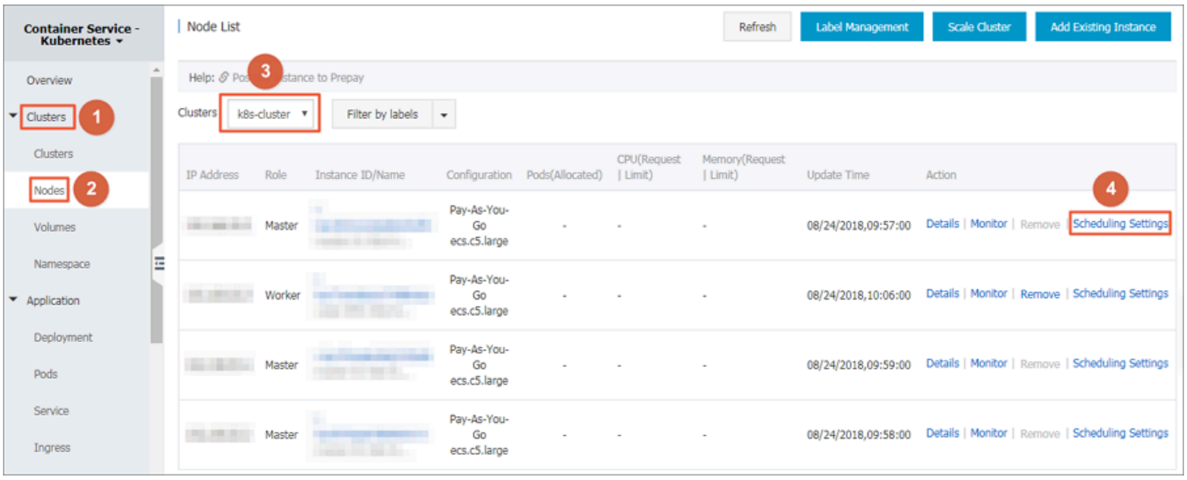
   Click Confirm in the displayed dialog box. The node label is deleted.

3.5 Set node scheduling

You can set node scheduling through the web interface so that you can allocate loads to each node properly.

Procedure

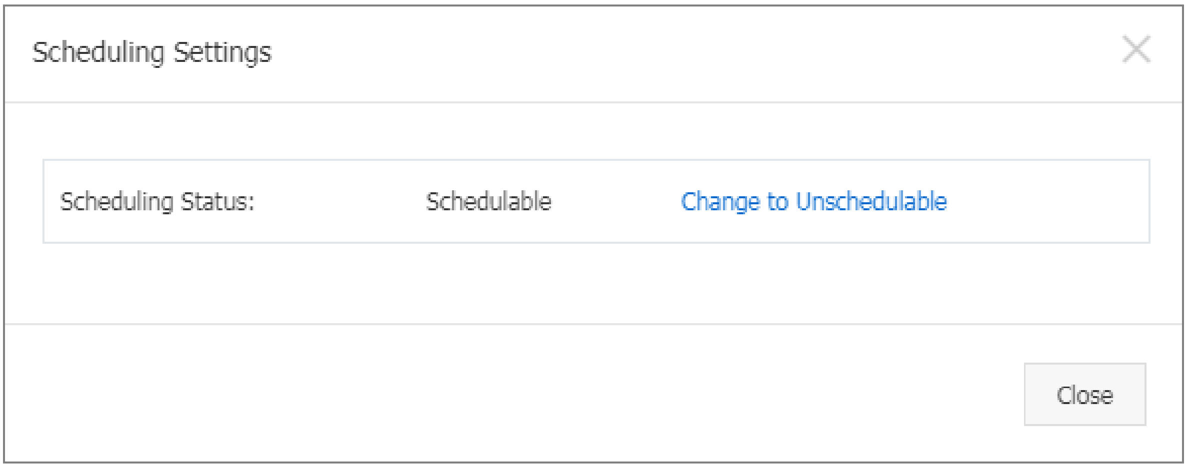
1. Log on to the Container Service console.
2. Under Kubernetes, click Clusters > Nodes to enter the Node List page.
3. Select a cluster, select a node under the cluster, and click Schedule Settings on the right.



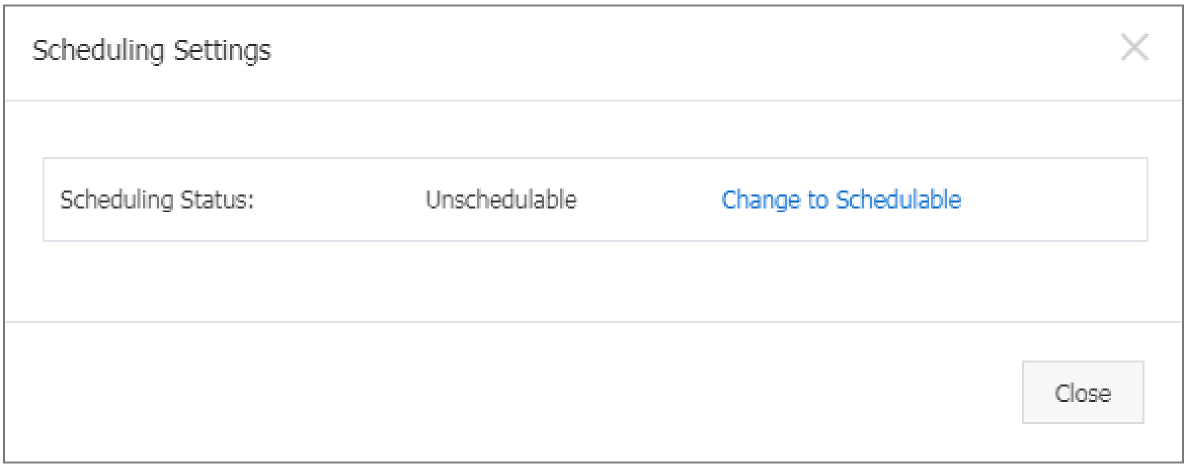
4. Set node scheduling in the displayed dialog box. In this example, click Change to Unschedulable to set the node to unschedulable.

Note:

The scheduling status of the current node is displayed in the Scheduling Settings dialog box, which is schedulable by default. You can change the status.



After the status is set, the scheduling status of the node changes in the dialog box.



What's next

When you deploy your application later, you can find that pods are not scheduled to the node.

3.6 Configure a Kubernetes GPU cluster to support GPU scheduling

From version 1.8, Kubernetes will support hardware acceleration devices such as NVIDIA GPU, InfiniBand, and FPGA, by using device plugins. Furthermore, GPU solutions of Kubernetes open source communities will be deprecated in version 1.10, and removed from the master code in version 1.11.

We recommend that you use an Alibaba Cloud Kubernetes cluster combined with GPU to run highly dense computational tasks such as machine learning and image processing. With this method, you can implement one-click deployment, elastic scaling, and other functions, without needing to install NVIDIA drivers or Compute Unified Device Architecture (CUDA) beforehand.

Background information

During cluster creation, Container Service performs the following operations:

- Creates Elastic Compute Service (ECS) instances, sets the public key used for SSH logon from the management node to other nodes, and installs and configures the Kubernetes cluster by using CloudInit.
- Creates a security group to allow inbound access to all ICMP ports in a VPC.
- Creates a new VPC and VSwitch if you do not use the existing VPC, and also creates an SNAT entry for the VSwitch.
- Creates VPC routing rules.
- Creates a NAT gateway and Elastic IP (EIP).
- Creates a Resource Access Management (RAM) user and AccessKey (AK). This RAM user has the permissions to query, create, and delete ECS instances, add and delete cloud disks, and all relevant access permissions for Server Load Balancer (SLB) instances, CloudMonitor, VPC, Log Service, and Network Attached Storage (NAS) services. The Kubernetes cluster dynamically creates the SLB instances, cloud disks, and VPC routing rules according to your configurations.
- Creates an intranet SLB instance and exposes port 6443.
- Creates an Internet SLB instance and exposes ports 6443, 8443, and 22. (If you enable the SSH logon for Internet access when creating the cluster, port 22 is exposed. Otherwise, port 22 is not exposed.)

Prerequisites

You have activated Container Service, Resource Orchestration Service (ROS), and RAM.

You have logged on to the Container Service console, ROS console, and RAM console to activate the corresponding services.

> **Note:**
> The deployment of Container Service Kubernetes clusters depends on the application deployment capabilities of Alibaba Cloud ROS. Therefore, you need to activate ROS before creating a Kubernetes cluster.

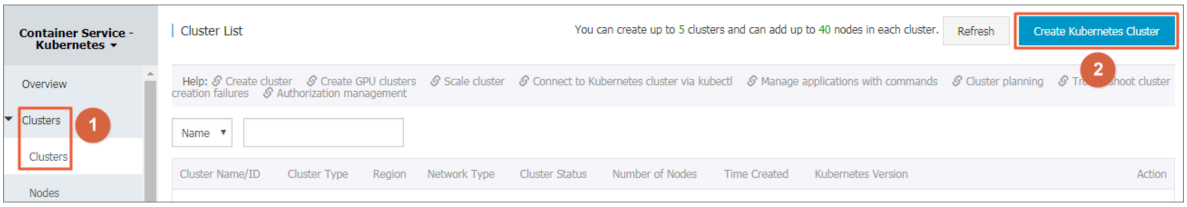
Limits

- The SLB instance created with the Kubernetes cluster only supports the Pay-As-You-Go billing method.
- The Kubernetes cluster supports only Virtual Private Cloud (VPC).
- By default, each account has a specified quota of the number of cloud resources that it can create. If the number of cloud resources has reached the quota limit, the account cannot create a cluster. Make sure you have sufficient resource quota to create a cluster. You can open a ticket to increase your quota.
  - By default, each account can create up to 5 clusters across all regions and add up to 40 nodes to each cluster. You can open a ticket to create more clusters or nodes.
  - By default, each account can create up to 100 security groups.
  - By default, each account can create up to 60 Pay-As-You-Go SLB instances.
  - By default, each account can create up to 20 EIPs.
- The limits for ECS instances are as follows:
  - Only the CentOS operating system is supported.
  - Only Pay-As-You-Go ECS instances can be created.

> Note:
> After creating an instance, you can Switch from Pay-As-You-Go to Subscription billing in the ECS console.

Create a GN5 Kubernetes cluster

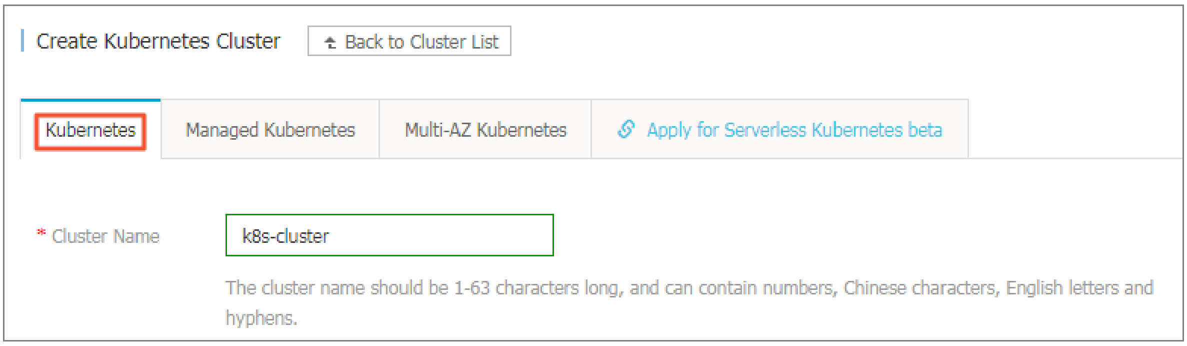
1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, click Clusters.
3. Click Create Kubernetes Cluster in the upper-right corner.



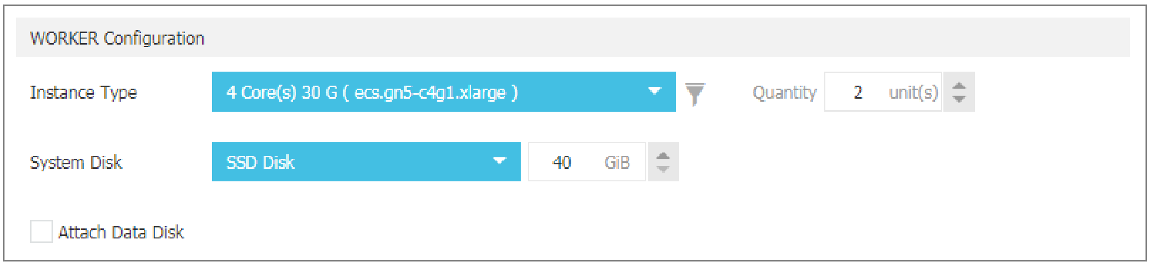
By default, the Create Kubernetes Cluster page is displayed.

> Note:

Worker nodes are set to use GPU ECS instances to create a GPU cluster. For information about other parameter settings, see Create a Kubernetes cluster.



4. Set the Worker nodes. In this example, the gn5 GPU instance type is selected to set Worker nodes as GPU working nodes.
   a. If you choose to create Worker instances, you must select the instance type and the number of Worker nodes. In this example, two GPU nodes are created.



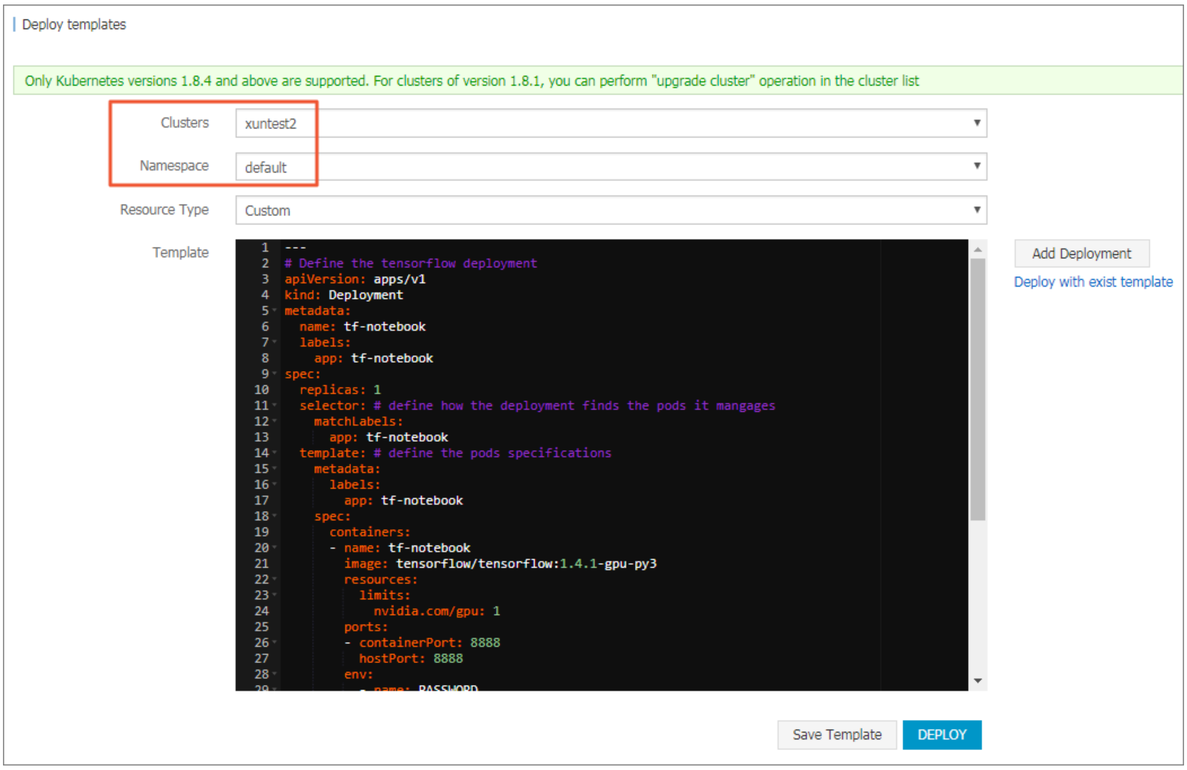
   b. If you choose to add existing instances, you need to have already created GPU cloud servers in the same region where the cluster is to be created.
5. After you have completed all required settings, click Create to start cluster deployment.
6. After the cluster is created, choose Clusters > Nodes in the left-side navigation pane.
7. To view the GPU devices mounted to either of the created nodes, select the created cluster from the clusters drop-down list, select one of the created Worker nodes, and choose More > Details in the action column.

Create a GPU experimental environment to run TensorFLow

Jupyter is a popular tool used by data scientists for the experimental environment TensorFlow. This topic describes an example of how to deploy a Jupyter application.

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Applications > Deployments.

3. Click Create by Template in the upper-right corner.
4. Select the target cluster and namespace and then select a sample template or the custom template from the resource type drop-down list. After you orchestrate your template, click DEPLOY.



In this example, a Jupyter application template is orchestrated. The template includes 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 manages
     matchLabel s :
       app : tf - notebook
   template : # define the pods specificat ions
     metadata :
       labels :
         app : tf - notebook
     spec :
       containers :
      - name : tf - notebook
         image : tensorflow / tensorflow : 1 . 4 . 1 - gpu - py3
         resources :
           limits :
```

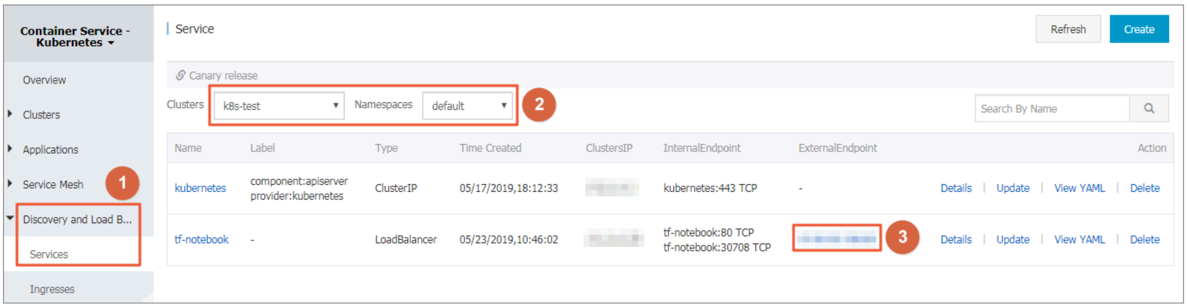
```
              nvidia . com / gpu :  1                       # specify
  the   number   of   NVIDIA   GPUs   that   are   called   by
 the   applicatio  n
          ports :
         -  containerP  ort :  8888
            hostPort :  8888
          env :
           -  name :  PASSWORD                            # specify
 the   password   used   to   access   the   Jupyter   service .
 You   can   modify   the   password   as   needed .
              value :  mypassw0rd

#  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 :  LoadBalanc  er                           # set   Alibaba
   Cloud   SLB   service   for   the   applicatio  n   so   that
 its   services   are   accessible   from   the   Internet .
```

If you use a GPU deployment solution of Kubernetes earlier than 1.9.3, you must define the following volumes in which the NVIDIA drivers reside:

```
 volumes :
     -  hostPath :
          path : / usr / lib / nvidia - 375 / bin
          name :  bin
     -  hostPath :
          path : / usr / lib / nvidia - 375
          name :  lib
```

When you orchestrate your deployment template in a cluster by using the GPU deployment solution of Kubernetes earlier than 1.9.3, your template must be highly dependent on the cluster. As a result, portability of the template is not achievable. However, in Kubernetes version 1.9.3 and later, you do not need to specify these hostPaths because the NIVEA plugins automatically discover the library links and execution files required by the drivers.

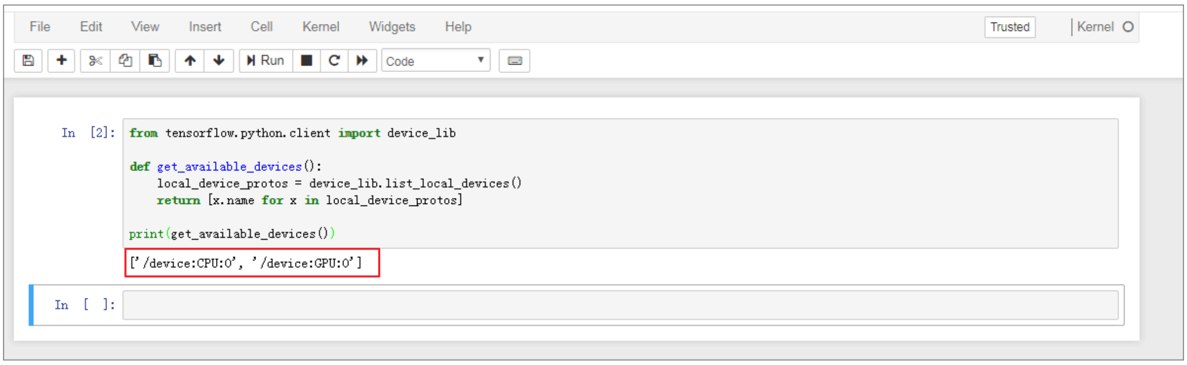
5. In the left-side navigation pane under Container Service-Kubernetes, choose Discovery and Load Balancing > Services. Then, select the target cluster and namespace, and then view the external endpoint of the tf-notebook service.



6. Access the Jupyter application in a browser. The access address is `http ://EXTERNAL - IP` . You need to enter the password set in the template.

7. By running the following program, you can verify that this Jupyter application can use GPU, and the program is able to list all devices that can be used by Tensorflow:

```
from   tensorflow . python . client   import   device_lib

def   get_availa  ble_device  s ():
    local_devi  ce_protos  =  device_lib . list_local  _devices ()
    return  [ x . name   for   x   in   local_devi  ce_protos ]

print ( get_availa  ble_device  s ())
```



3.7 Remove a node

Before you restart or release an ECS instance in a Kubernetes cluster, you need to remove the ECS node from the cluster. This topic describes how to remove a node from a Kubernetes cluster.

Prerequisites

- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.

- You have connected to the Kubernetes cluster by using kubectl, see Connect to a Kubernetes cluster by using kubectl.

Context

- Removing a node causes pod migration. This may affect the services provided by the pods running on the node. Therefore, we recommend that you remove a node only when fewer services are in demand.
- Removing a node may cause unintended risks. We recommend that you back up your data in advance and exercise caution when performing this action.
- When you start to remove a node, the node is automatically set to the unschedula ble status.
- Only Worker nodes can be removed.

Procedure

1. Run the following command to migrate the pods on the target node to other nodes:

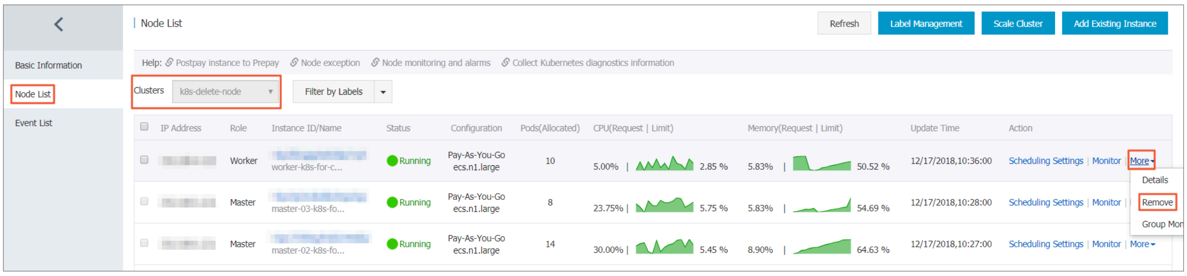
   Note:
   You must ensure that other nodes in the Kubernetes cluster have sufficient resources to run the pods that you want to migrate.

   ```
   kubectl drain node-name
   ```

   Note:
   The `node-name` parameter must be in the format of *`your-region-name.node-id`*.

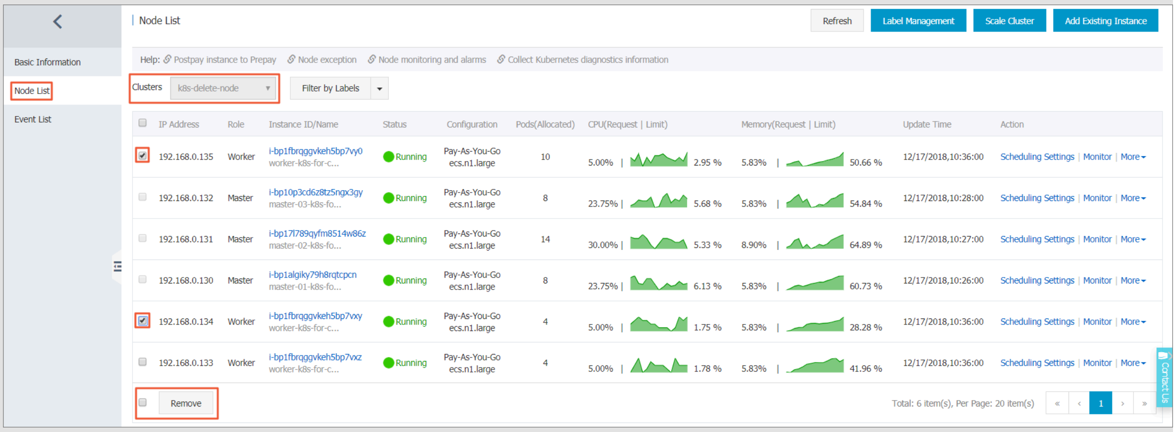
   - *`your-region-name`* indicates the name of the region where your cluster resides.
   - *`node-id`* indicates the ID of the ECS instance in which the node to be removed resides. For example, `cn - hanghzou . i - xxx .`
2. In the left-side navigation pane under Kubernetes, choose Clusters > Nodes.

3. Under the target cluster, select the target node, and choose More > Remove in the Action column.

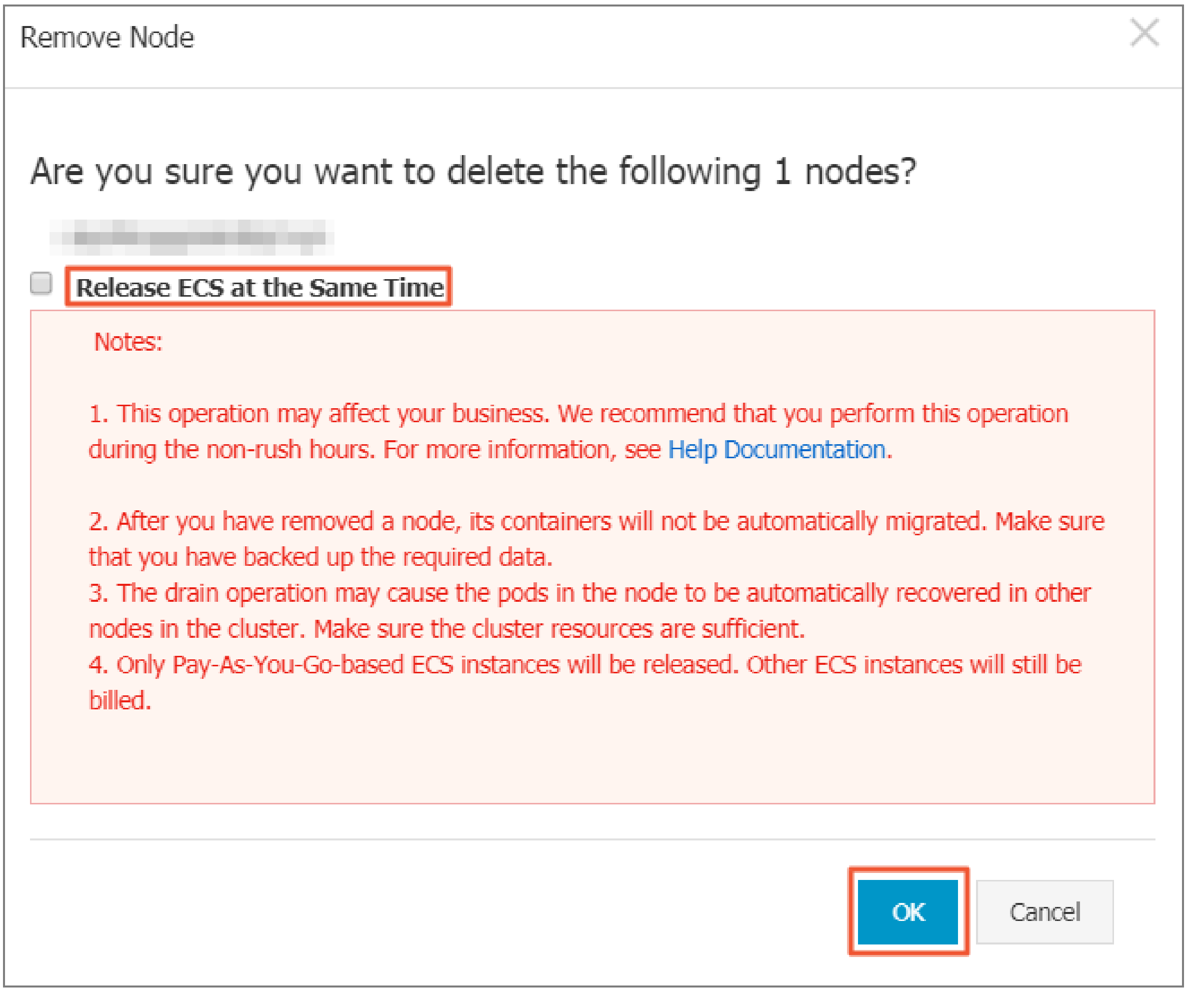


Note:

If you want to remove multiple nodes at a time, you can select the target cluster on the Node List page, select all the nodes to be removed, and then click Remove.



4. Optional: Select the Release ECS at the Same Time check box to permanently release the ECS instance where the node resides.



Note:

- Only Pay-As-You-Go ECS instances can be released.
- A Subscription ECS instance will be released automatically when it expires.
- If you do not select the Release ECS at the Same Time check box, the ECS instance in which the node resides will continue to be charged.

5. Click OK.

3.8 Use Alibaba Cloud Kubernetes GPU node labels for scheduling

When you implement GPU computing through a Kubernetes cluster, you can schedule an application to the node installed with GPU devices as needed by using GPU node labels.

Prerequisites

- You have created a Kubernetes cluster that has GPU nodes. For more information, see Configure a Kubernetes GPU cluster to support GPU scheduling.
- You have connected to the Master node, which makes it easier to view node labels and other information. For more information, see Connect to a Kubernetes cluster by using kubectl.

Context

When deploying NVIDIA GPU nodes, Kubernetes that runs on Alibaba Cloud discovers the GPU attribute and exposes it as the node label information. Node labels provide the following benefits:

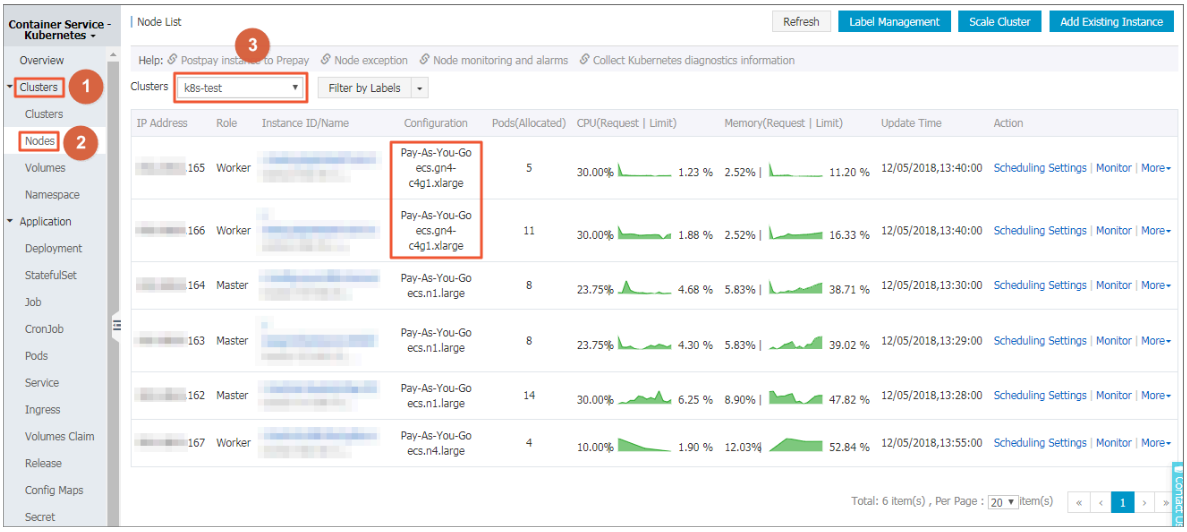
1. Node labels help you filter GPU nodes.
2. Node labels can be used as the scheduling conditions for application deployment.

Procedure

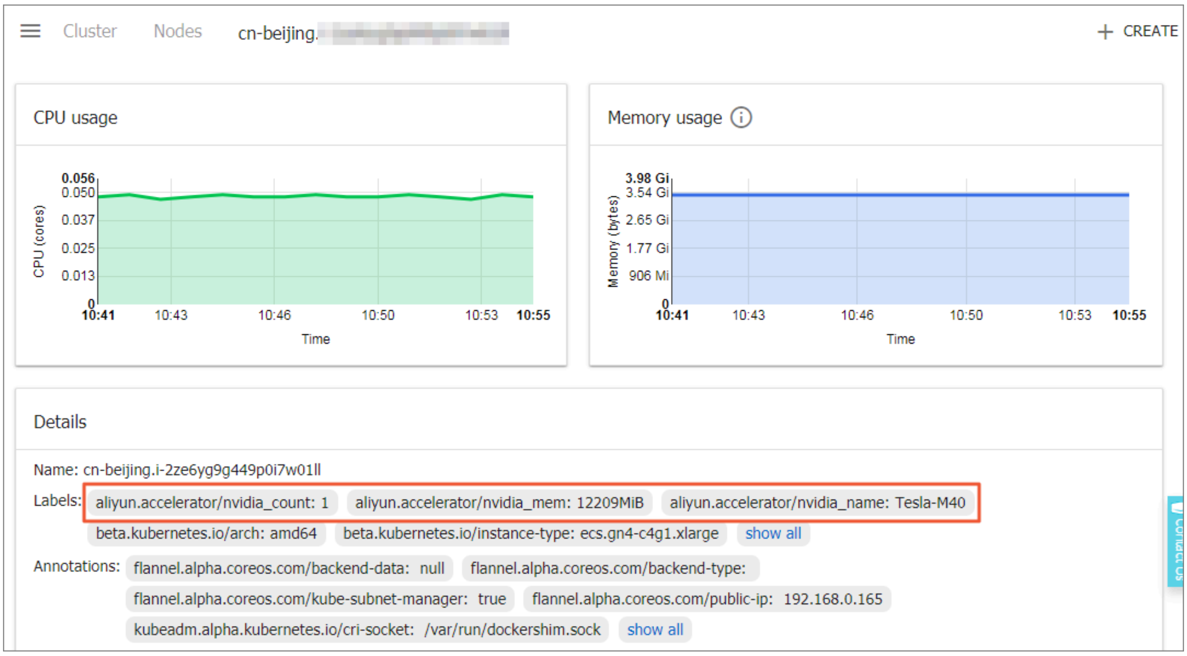
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Nodes.

   > Note:

In this example, the cluster has three Worker nodes of which two Worker nodes are mounted with GPU devices. You need to view the node IP addresses for verification.



3. Select a GPU node, and choose More > Details in the action column. Then, you can view the GPU node label on the Kubernetes dashboard.



You can also log on to a Master node and run the following command to view the GPU node label:

```
#  kubectl   get   nodes
 NAME                                  STATUS      ROLES        AGE
      VERSION
 cn - beijing . i - 2ze2dy2h9w  97v65uuaft     Ready       master
   2d          v1 . 11 . 2
 cn - beijing . i - 2ze8o1a45q  dv5q8a7luz     Ready       < none >
     2d           v1 . 11 . 2            # Compare   this   node
```

```
with   the   node   displayed   in   the   console   to   determine
  the   GPU   node .
cn - beijing . i - 2ze8o1a45q  dv5q8a7lv0     Ready      < none >
  2d         v1 . 11 . 2
cn - beijing . i - 2ze9xylyn1  1vop7g5bwe     Ready       master
  2d         v1 . 11 . 2
cn - beijing . i - 2zed5sw8sn  jniq6mf5e5     Ready       master
  2d         v1 . 11 . 2
cn - beijing . i - 2zej9s0zij  ykp9pwf7lu     Ready      < none >
  2d         v1 . 11 . 2
```

Select a GPU node and run the following command to view the GPU node label:

```
#  kubectl   describe   node   cn - beijing . i - 2ze8o1a45q
 dv5q8a7luz
 Name :                cn - beijing . i - 2ze8o1a45q  dv5q8a7luz
 Roles :              < none >
 Labels :              aliyun . accelerato  r / nvidia_cou  nt = 1
                         # This   field   is   important .
                     aliyun . accelerato  r / nvidia_mem = 12209MiB
                     aliyun . accelerato  r / nvidia_nam  e = 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 -
 2ze8o1a45q  dv5q8a7luz
 ......
```

In this example, the GPU node contains the following three node labels:

Key	Value
`aliyun . accelerato  r / nvidia_cou  nt`	Number of GPU cores
`aliyun . accelerato  r / nvidia_mem`	GPU memory in MiB
`aliyun . accelerato  r / nvidia_nam  e`	Name of the GPU computing card of the NVIDIA device

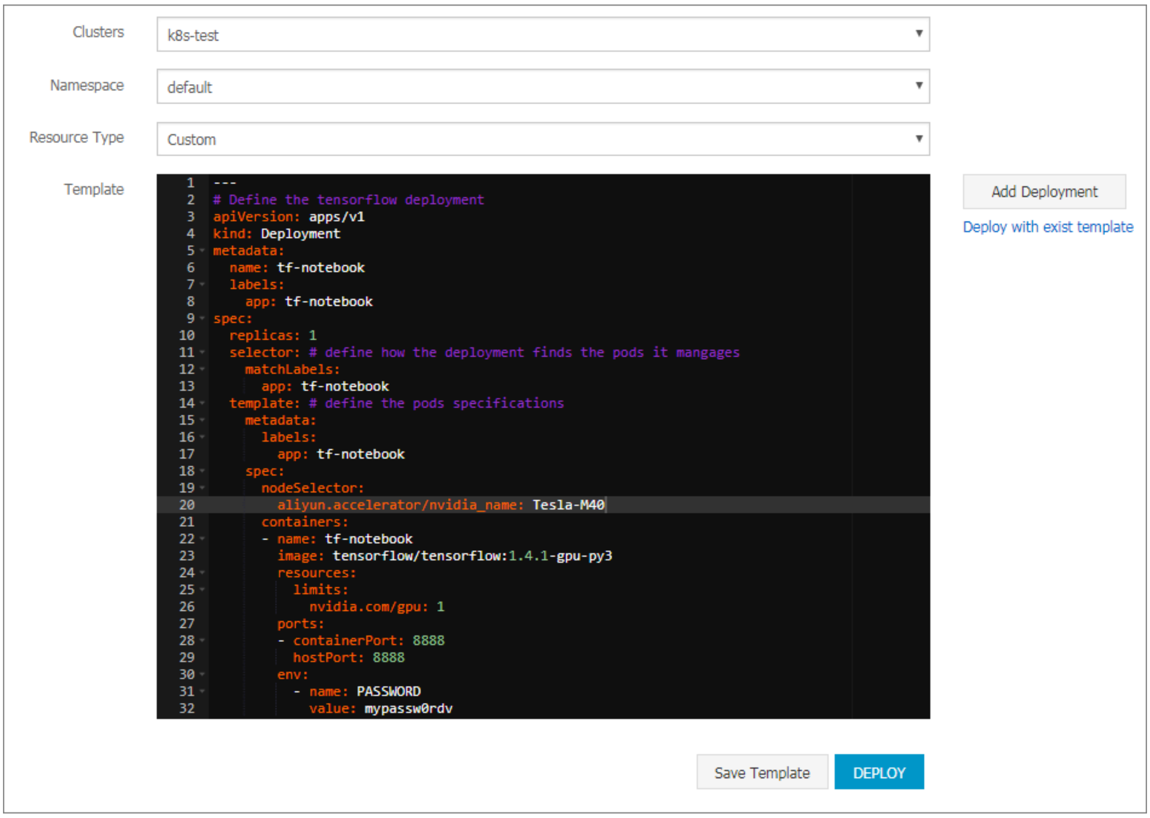
The GPU cloud servers of the same type share the same GPU computing card name . Therefore, you can use this label to filter nodes.

```
#  kubectl   get   no  - l   aliyun . accelerato  r / nvidia_nam  e
 = Tesla - M40
 NAME                                    STATUS      ROLES       AGE
       VERSION
 cn - beijing . i - 2ze8o1a45q  dv5q8a7luz     Ready      < none >
   2d         v1 . 11 . 2
```

```
cn - beijing . i - 2ze8o1a45q  dv5q8a7lv0     Ready      < none >
  2d           v1 . 11 . 2
```

4. Return to the Container Service console home page. Then, in the left-side navigation pane, choose Applications > Deployments, and click Create by Template in the upper-right corner.

   a) Create a TensorFlow application and schedule this application to the GPU node.



In this example, the YAML template is orchestrated as follows:

```
---
#  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   manages
     matchLabel  s :
       app :  tf - notebook
   template : # Define   the   pod   specificat  ions .
     metadata :
       labels :
         app :  tf - notebook
     spec :
       nodeSelect  or :
       # This   field   is   important .
```

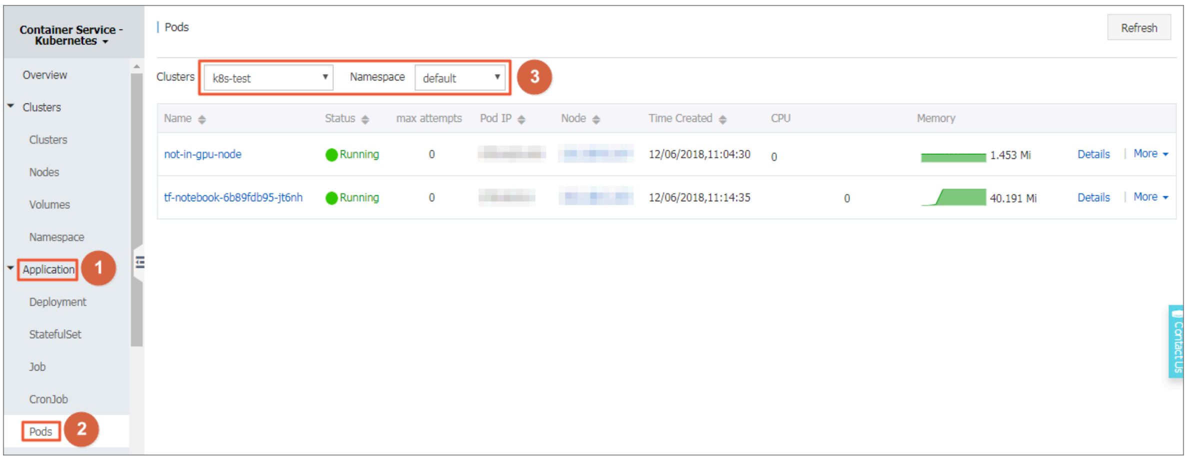
```
        aliyun . accelerato  r / nvidia_nam  e :  Tesla - M40
      containers :
     - name :  tf - notebook
        image :  tensorflow / tensorflow : 1 . 4 . 1 - gpu - py3
        resources :
          limits :
            nvidia . com / gpu :  1
          # This   field   is   important .
        ports :
        - containerP  ort :  8888
          hostPort :  8888
        env :
         - name :  PASSWORD
           value :  mypassw0rd  v
```

b) You can also avoid deploying an application to a GPU node. The following deploys an Nginx pod and schedules it by using the node affinity feature. For more information about node affinity, see Create a deployment application by using an image.

The example YAML template is orchestrated as follows:

```
apiVersion :  v1
kind :  Pod
metadata :
  name :  not - in - gpu - node
spec :
  affinity :
    nodeAffini  ty :
      requiredDu  ringSchedu  lingIgnore  dDuringExe  cution :
        nodeSelect  orTerms :
       - matchExpre  ssions :
         - key :  aliyun . accelerato  r / nvidia_nam  e
           operator :  DoesNotExi  st
  containers :
 - name :  not - in - gpu - node
    image :  nginx
```

5. In the left-side navigation pane, choose Applications > Pods, and select the target cluster and namespace.



Result

In the pod list, you can see that the two example pods have been scheduled to the target nodes, indicating you have implemented flexible scheduling by using GPU node labels.

3.9 View resource request and limit on nodes

The Container Service Console allows you to view resource usage of each node in a Kubernetes cluster.

Prerequisites

You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.

Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Clusters > Nodes.

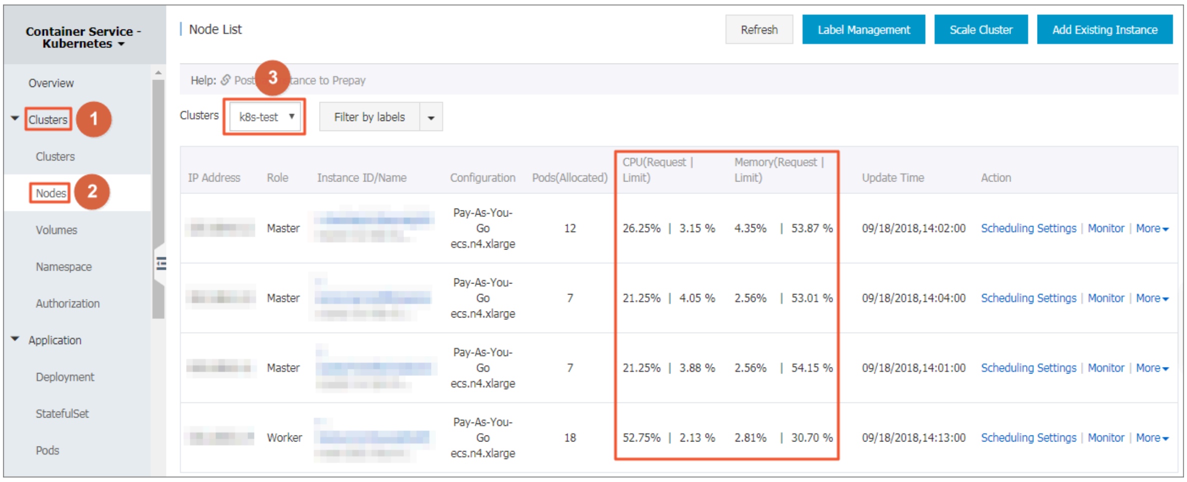
   You can view the resource usage for the CPU and memory of each node, namely, the request and limit, which are calculated as follows:

   - CPU request = sum (CPU request value from all pods on the current node) /total CPU of the current node.
   - CPU limit= sum (actual CPU usage of all pods on the current node)/total CPU of the current node.
   - Memory request = sum (memory request value from all pods on the current node) /total memory of the current node.
   - Memory limit= sum (actual memory usage of all pods on the current node)/total memory of the current node.

   Note:

   - You can allocate loads to a node based on the resource usage on the node. For more information, see Set node scheduling.

- When both the request and limit on a node is 100%, no new pod is scheduled to the node.



3.10 Mount a disk to a Kubernetes cluster node

This topic describes how to mount a disk to a Kubernetes cluster node. Mounting a disk allows you to expand the Docker data directory and maintain a sufficient disk capacity when the number of containers or images that run on a node increases.

Prerequisites

Your Kubernetes cluster version must be v1.10.4 or later.

You can mount a disk to an existing Kubernetes cluster node by using either of the following methods:

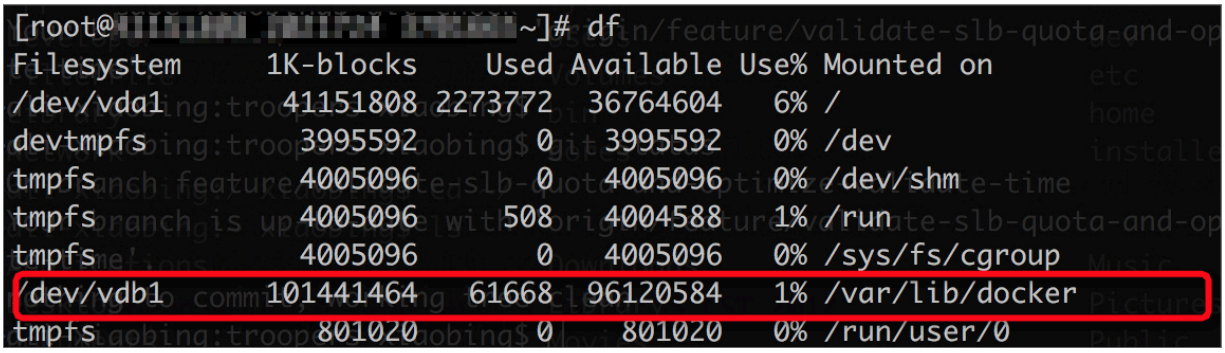
- If no disk is mounted to the existing node, see Mount a disk to the Docker data directory.
- If you have created a disk for the existing node, but you have failed to mount the disk to the node, you can follow these steps.

Note:

- We recommend that you create a snapshot of the target node or back up node data to avoid data loss.
- Additionally, you must ensure that you can schedule your cluster applications to other nodes.
- We recommend that you perform this operation during off-peak service hours to avoid disruptions to your business.

- Draining a node reschedules pods on the node to other nodes. Therefore,you must ensure that your Kubernetes cluster contains sufficient nodes.We recommend that youadd cluster nodes in advance as needed.

Before performing the operation, you need to determine whether a disk is already mounted to the target cluster node. To do so, run the `df` command on the target Worker node, and then check whether `/ var / lib / docker` has been mounted to `/ dev / vdb1`. If the disk mounting operation failed, you can mount the disk by following these 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
```

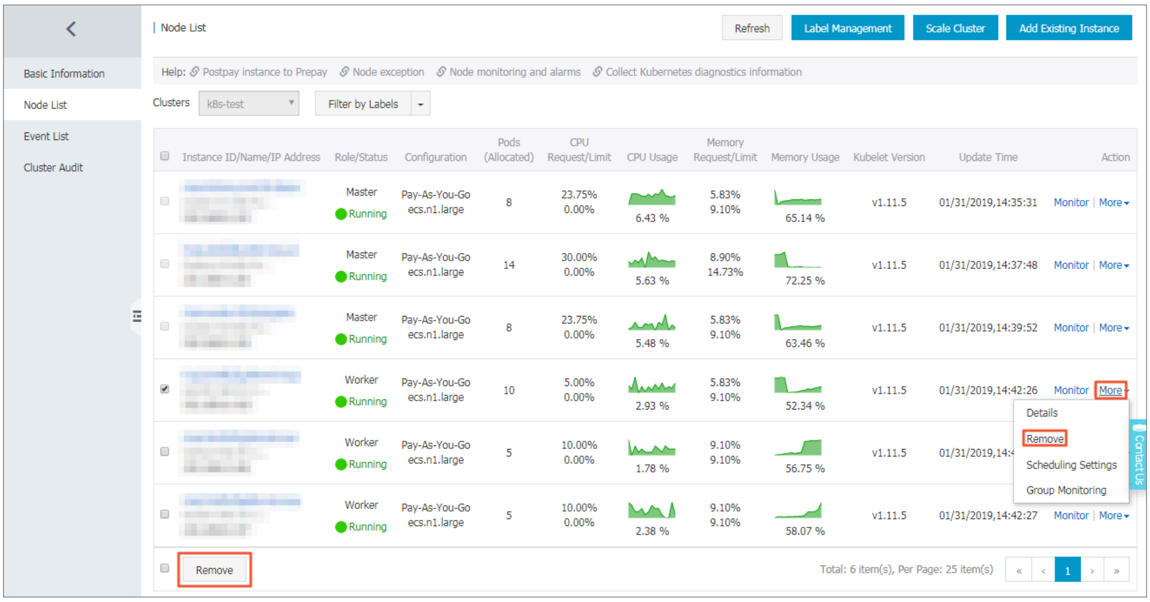
1. Set the target node as unschedulable. For more information, see Mark node as unschedulable.
2. Drain the target node. For more information, see Safely drain a node.

3. Remove the target node. This topic uses the Container Service console as an example.

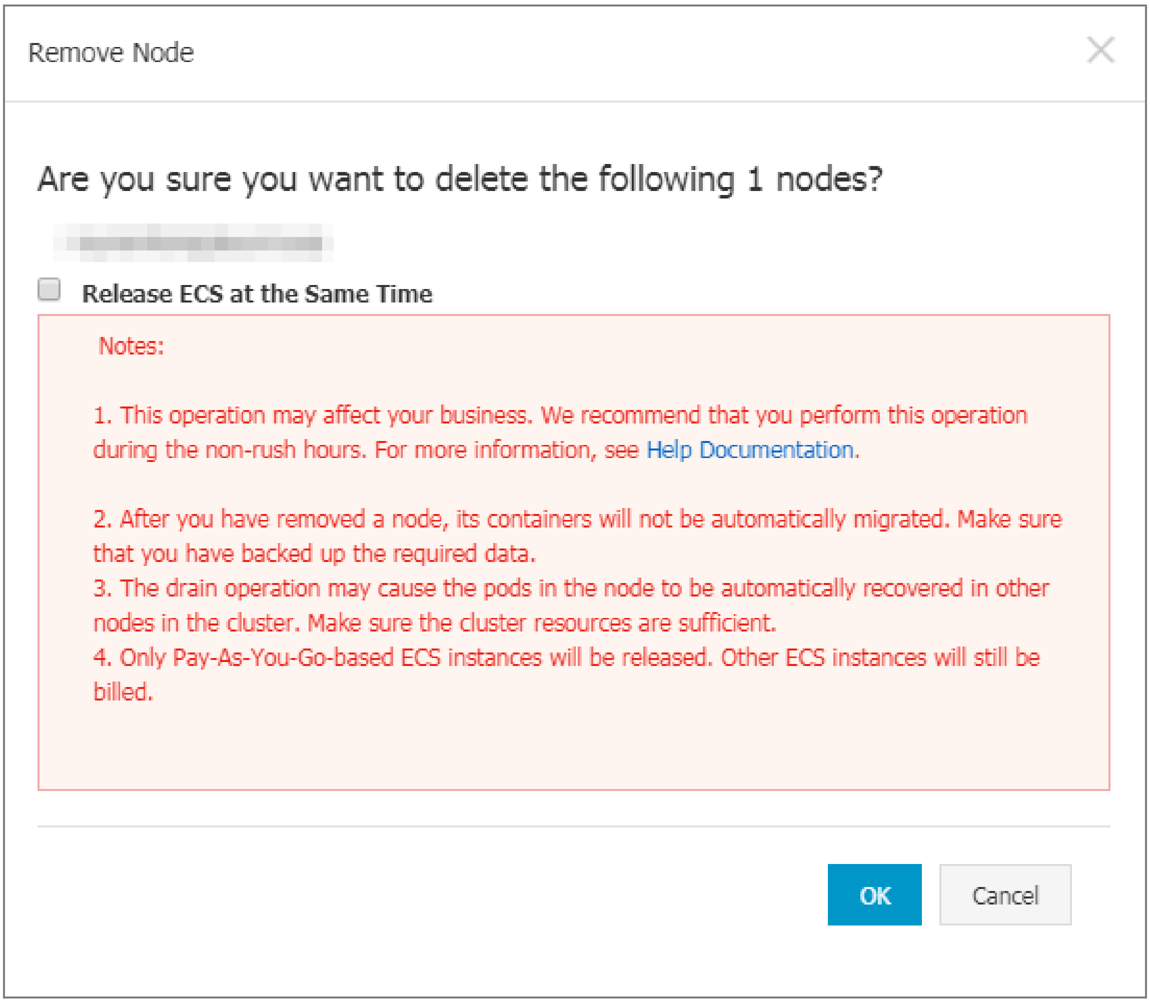
   a. Log on to the Container Service console.

   b. In the left-side navigation pane, click Node.

   c. Select the target node, and click Remove or choose More > Remove.



   d. In the displayed Remove Node dialog box, click OK.



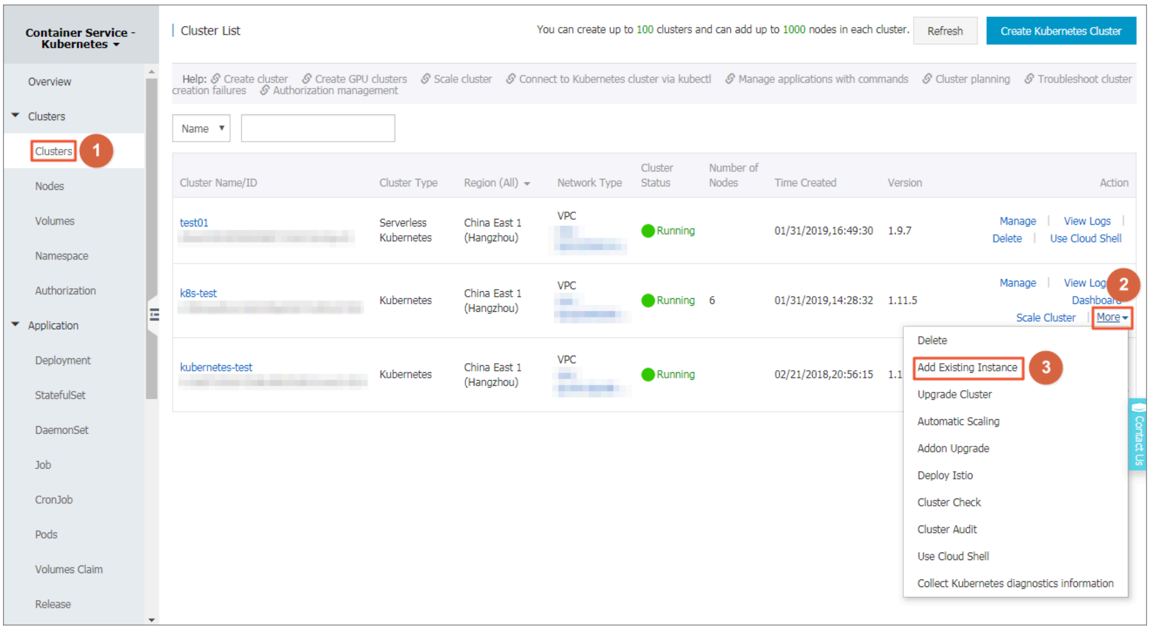
Note:

We recommend that you do not select the Release ECS at the same time check box. Otherwise, the ECS instance used by the target node will be released.

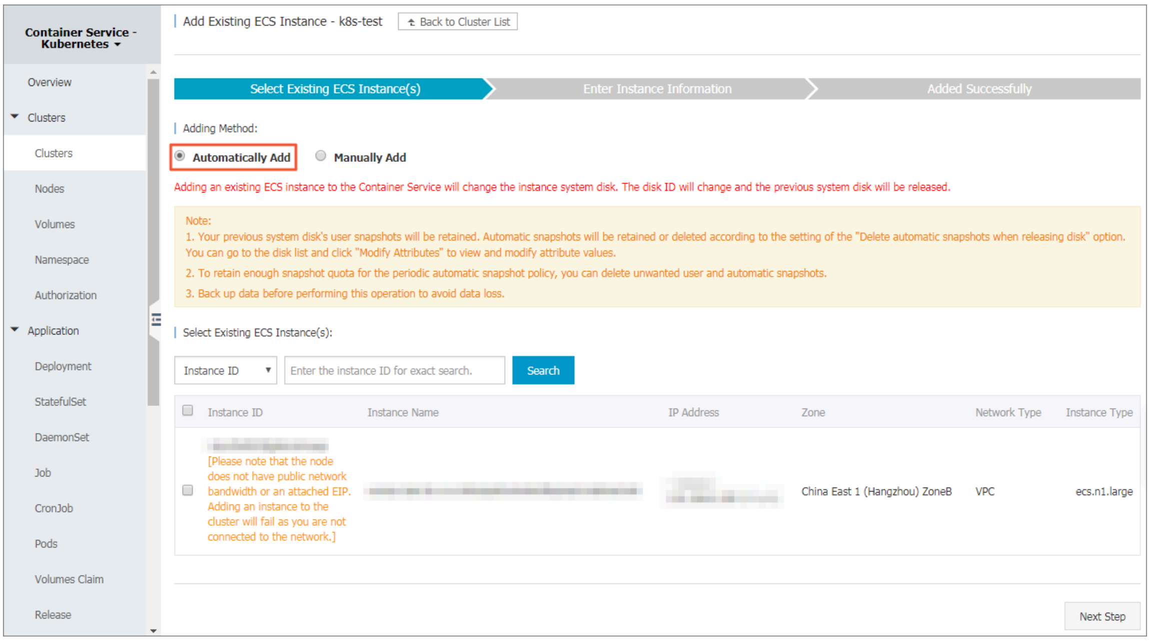
4. Add the removed node to the cluster.

   a. In the left-side navigation pane, click Clusters.

   b. On the right of the target cluster, choose More > Add Existing Instance.

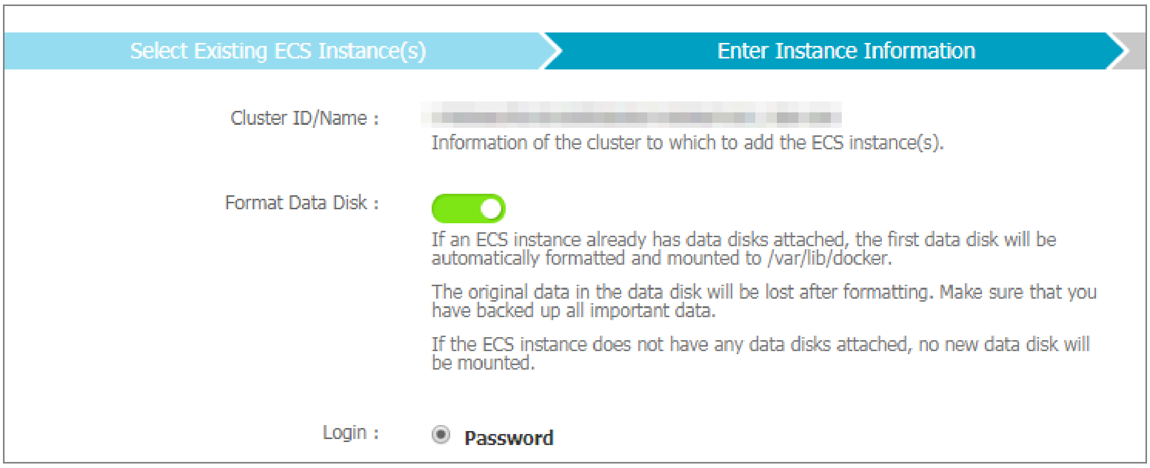


   c. Select Automatically Add or Manually Add. In this example, the instance is added automatically.



   d. Select the existing instance and then click Next Step.

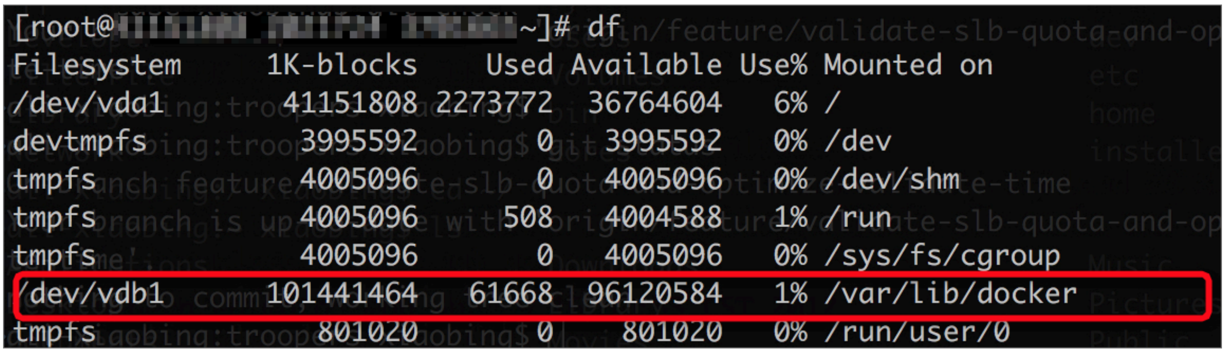
   e. Turn on the Format Data Disk switch.



f. Complete other required settings.

After the node has been added to the cluster, you can log on to the node to run the `df` command to check whether a disk has been mounted to the target node.

The following figure shows the disk has been amounted to the target node.



3.11 Mount a disk to the Docker data directory

This topic describes how to mount a disk to the Docker data directory. If the number of containers or images that run on an ECS instance increases constantly, the ECS instance disk capacity may be insufficient. In this case, you can expand the Docker data directory by mounting a disk to the ECS instance.

Docker data directory

Docker data is stored in disks through a union file system (UnionFS). 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 size occupied by this directory.

```
# du - h -- max - depth = 0 / var / lib / docker
```

```
7 . 9G      / var / lib / docker
```

Scenarios

Generally, a Docker image occupies a large amount of disk space. If you want to use multiple Docker images or a large number of containers, you must mount a disk to the Docker data directory to ensure sufficient disk capacity is available.

Mount a disk

To mount a disk to the Docker data directory, follow these steps:

1. Create a disk and mount it to the target ECS instance for which you want to expand the disk capacity.
   a. Log on to the ECS console to create a disk.
   b. In the left-side navigation pane, click Instances.
   c. Click the target ECS instance ID.
   d. In the left-side navigation pane, click Disks.
   e. In the upper-right corner, click Mount.
   f. In the displayed dialog box, select the created disk from the target disk drop-down list, and then click OK.
   g. Click Mount to mount the new disk to the target ECS instance, and record the new disk mounting point which is in the format of `/ dev / xvd *` or `/ dev / vd *`.

2. Log on to the target ECS instance to format the new disk.

   a. Run the `ls -l /dev/xvd*` or `ls -l /dev/vd*` command to verify whether a disk that has the recorded mounting point has been mounted to the ECS instance.

   b. Run the `fdisk` command to partition the new disk, and then run the `mkfs.ext4` command to format the new 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
```

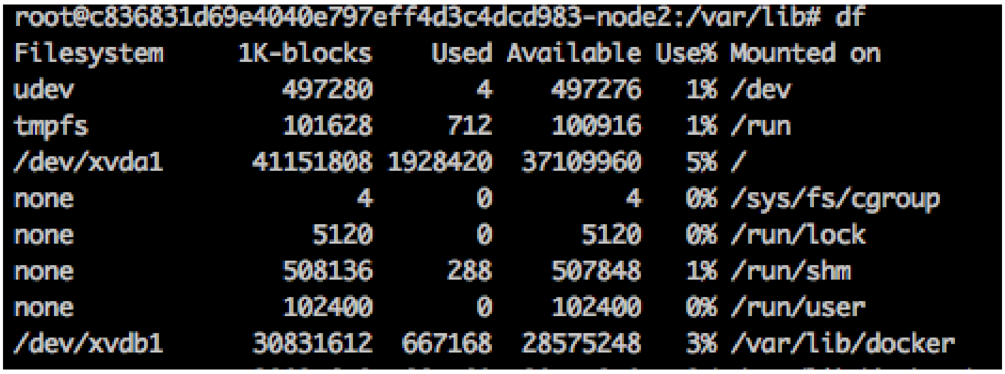
3. Migrate the Docker data to the new disk.

   If you do not want to suspend the applications that run on the target ECS instance, you must migrate the applications. For how to migrate applications on a Swarm cluster, see Schedule an application to specified nodes. For how to migrate applications on a Kubernetes cluster, see Safely drain a node while respecting application SLOs.

   a. To ensure that data can be migrated, run the `service docker stop` command to stop Docker daemon, and run the `service kubelet stop` command to stop kubelet.
   b. Migrate the Docker directory data to a backup directory. For example, `mv /var/lib/docker /var/lib/docker_data .`
   c. Mount the new disk to the */var/lib/docker* and */var/lib/kubelet* directories. For example,

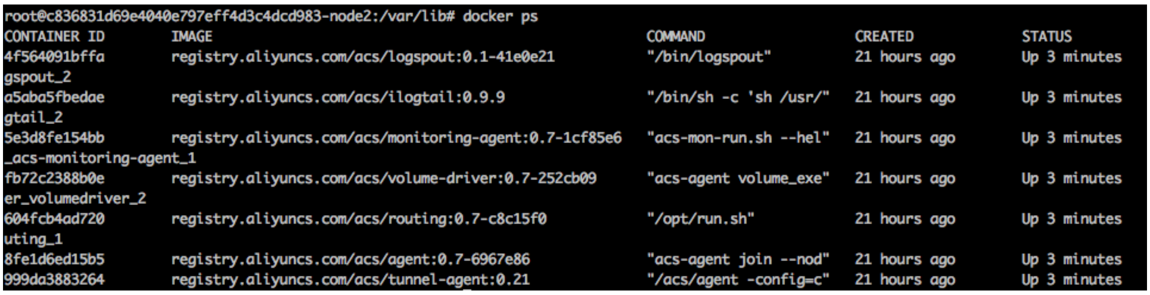
   ```
   echo "/dev/xvdb1    /var/lib/container/     ext4
   defaults        0    0" >>/etc/fstab
   echo "/var/lib/container/kubelet  /var/lib/kubelet
   none  defaults,bind  0  0" >>/etc/fstab
   echo "/var/lib/container/docker  /var/lib/docker
   none  defaults,bind  0  0" >>/etc/fstab


   mkdir /var/lib/docker

   mount -a
   ```
   d. Migrate the backed up Docker data to the new disk. For example, `mv /var/lib/docker_data/* /var/lib/docker/.`

4. Start the Docker daemon and kubelet, and check the data location.
   a. Run the `service docker start` command to start the Docker daemon, and run the `service kubelet start` command to start kubelet.
   b. Run the `df` command to verify whether `/ var / lib / docker` has been mounted to the new disk. If you need 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
```

   c. Run the `docker ps` command to check whether containers are lost. Restart containers as needed. For example, you can restart a container that has not been set 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 target node to which you mounted the new disk.

For more information, see Container Service.

4 Application management

4.1 Create a deployment application by using an image

This topic describes how to use an image to create a deployment application. In this topic, an Nginx application that is accessible to the Internet is created.

Prerequisites

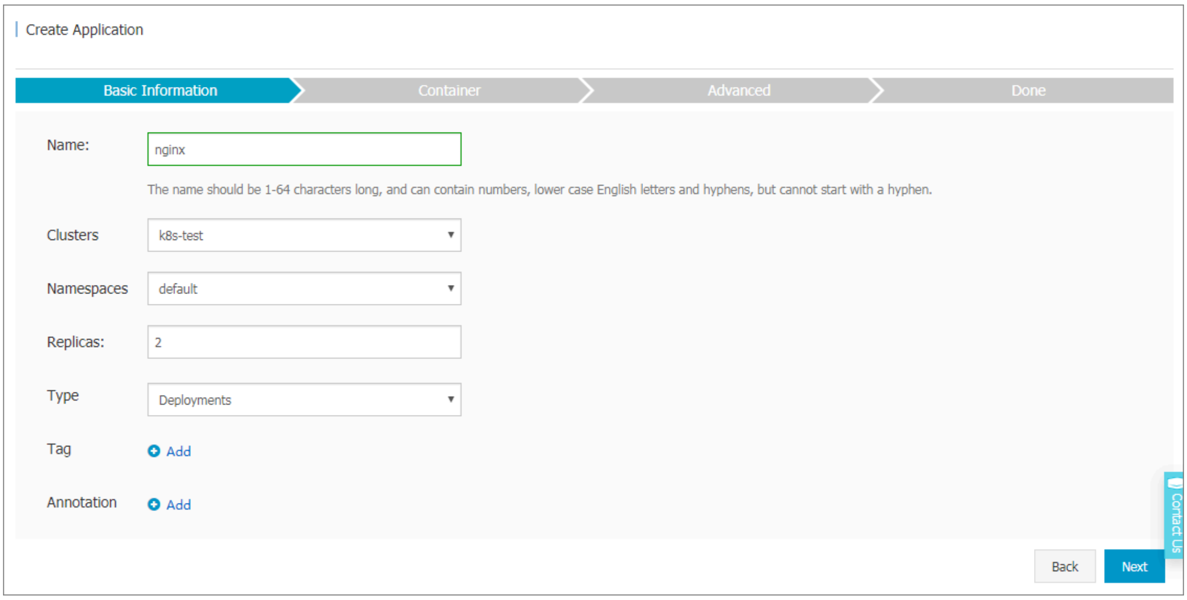
A Kubernetes cluster is created with ACK. For more information, see Create a Kubernetes cluster.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Application > Deployment, and then click Create by Image in the upper-right corner.
3. Set Name, Cluster, Namespace, Replicas, Type, Tag, and Annotation. The replicas parameter indicates the number of pods contained in the application. Then click Next.

   Note:
   In this example, you need to select the Deployment type.

   If you do not set Namespace, the system automatically uses the default namespace.



4. Configure a container.

> **Note:**
> You can configure multiple containers for the pod of the application.

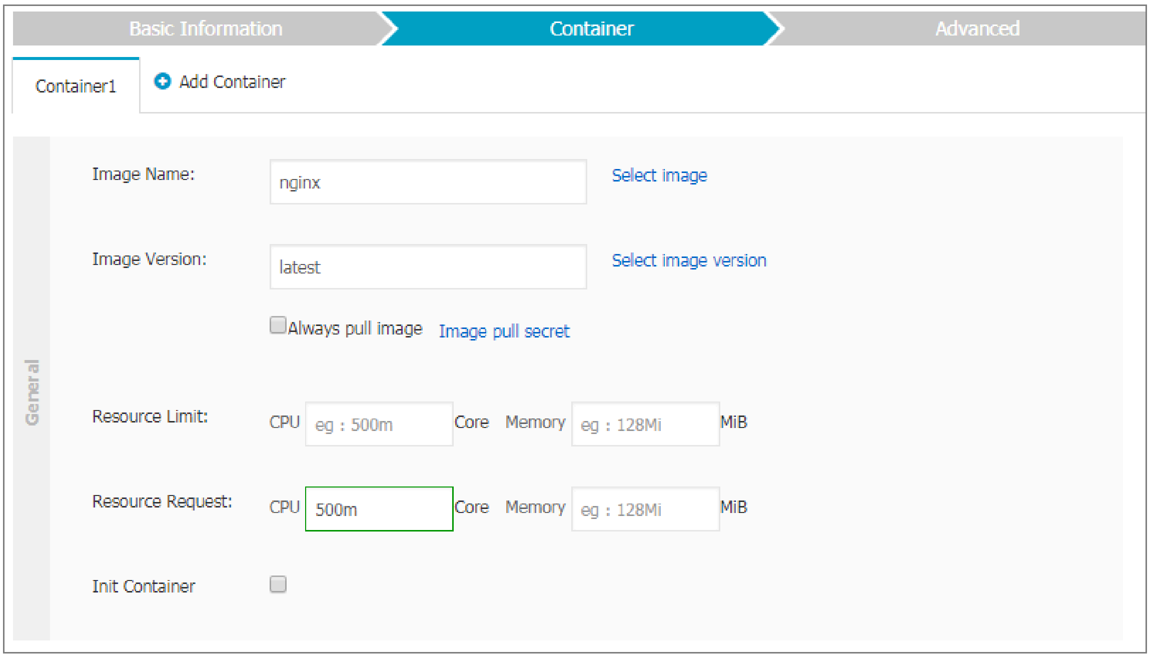
a) Set general container parameters.

- Image Name: Click Select image to select the image in the displayed dialog box and then click OK. In this example, select the Nginx image.

  You can also enter a private registry in the format of `domainname / namespace / imagename : tag` to specify an image.
- Image Version: Click Select image version to select a version. If you do not select an image version, the system uses the latest version by default.
- Always pull image: Container Service caches the image to improve deployment efficiency. During deployment, if the tag of the newly specified image is the same as that of the cached image, Container Service reuses the cached image, instead of re-pulling the same image. Therefore, if you do not modify the image tag when changing your code and image, the early image in the local cache is used in the application deployment. If you select this check box, Container Service ignores the cached image and re-pulls an image when deploying the application to make sure the latest image and code are always used.
- Image pull secret: Create a Secret for the image. A secret is required to pull a image from a private image repository. For more information, see Use an image Secret.
- Resource Limit: Specify the upper limit for the resources (CPU and memory) that can be used by this application to avoid occupying excessive resources. CPU is measured in the number of cores. Memory is measured in bytes, which can be MiB.
- Resource Request: Specify how many resources (CPU and memory) are reserved for the application. These resources can be set to be exclusive to the container by using this parameter. If you do not set this parameter, other

services or processes will compete for resources. Then the application may become unavailable due to resource shortage.

- Init Container: Select this check box to create an Init Container that contains useful tools. For more information, see Init containers.

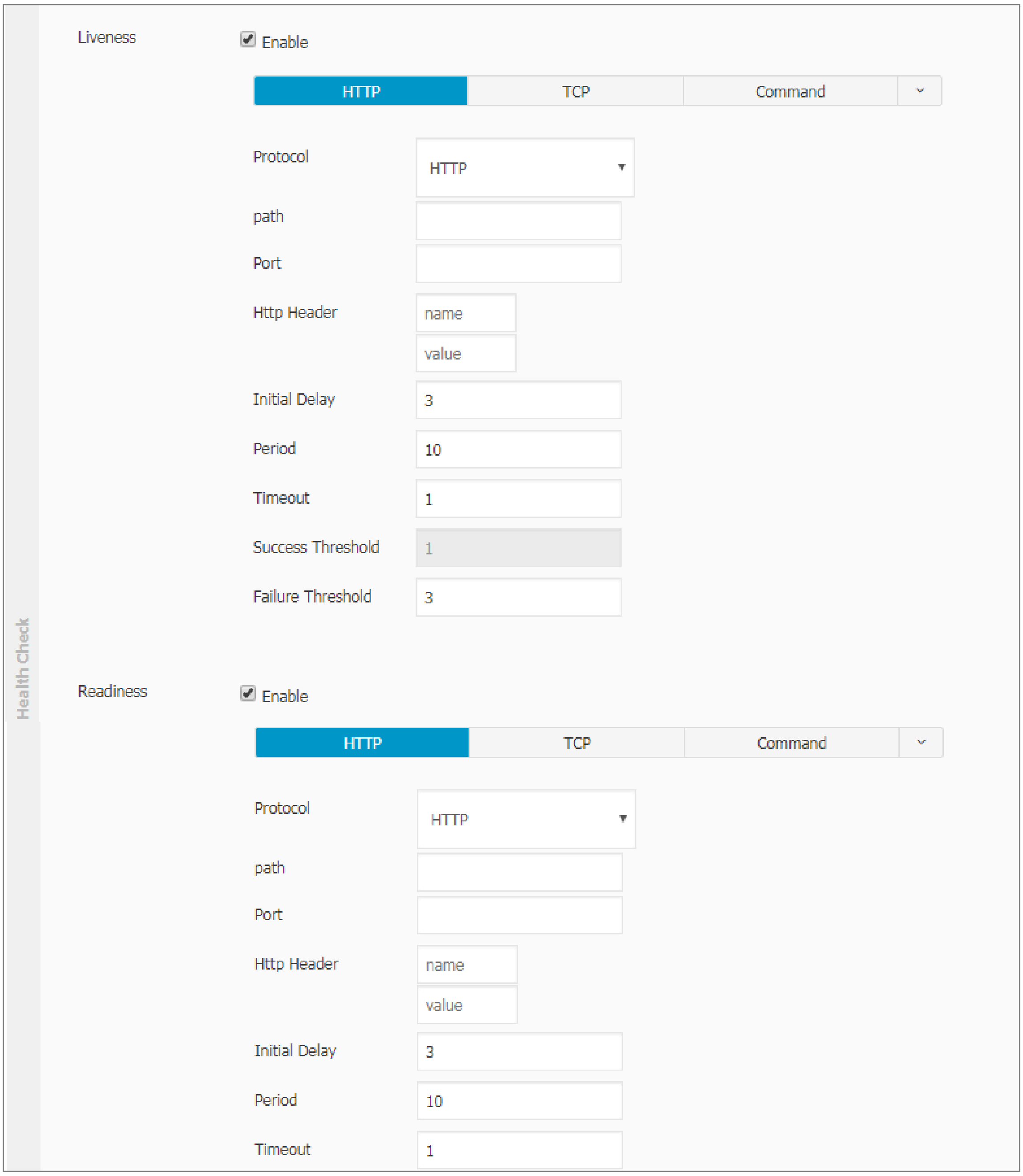


b) Optional: Set environment variables.

You can use key-value pairs to set environment variables for the pods. Environment variables are used to add environment labels or pass configurations for the pods. For more information, see Pod variable.

c) Optional: Set health checks.

You can set liveness probes and readiness probes. Liveness probes are used to detect when to restart the container. Readiness probes determine if the container is ready to receive traffic. For more information about health checks, see Configure liveness and readiness probes.



Request method	Description
HTTP request	With this health check method, you can send an HTTP GET request to the container. The following parameters are supported: · Protocol: HTTP/HTTPS. · Path: the path to access the HTTP server. · Port: the number or name of the access port exposed by the container. The port number must be in the range of 1 to 65535. · HTTP Header: custom headers in the HTTP request. HTTP allows repeated headers. You can use a key-value pair to set an HTTP Header. · Initial Delay (in seconds): the initialDelaySeconds parameter, indicating the number of seconds for which the first probe must wait after the container is started. The default value is 3. · Period (in seconds): the periodseconds parameter, indicating the interval at which probes are performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): the timeoutSeconds parameter, indicating the number of time that the probe has timed out. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes needed for determining a probe success after a failed probe. The default value is 1 and the minimum value is 1. It must be set to 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes needed for determining a probe failure after a successful probe. The default value is 3. The minimum value is 1.

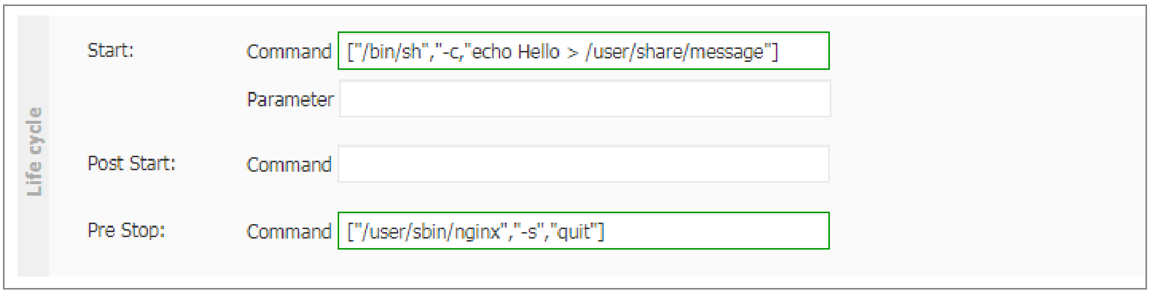
Request method	Description
TCP connection	If you use this health check method, a TCP socket is sent to the container. The kubelet then attempts to open the socket of the container on a specified port. If a connection can be established, the container is considered healthy. If not, it is considered unhealthy. The following parameters are supported: · Port: the number or name of the access port exposed by the container. The port number must be in the range of 1 to 65535. · Initial Delay (in seconds): the initialDelaySeconds parameter, indicating the seconds for the first liveness or readiness probe must wait for after the container is started. The default value is 15. · Period (in seconds): the periodseco nds parameter, indicating the interval at which probes are performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): the timeoutSeconds parameter, indicating the number of time that the probe has timed out. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes needed for determining a probe success after a failed probe . The default value is 1 and the minimum value is 1. It must be set to 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes needed for determining a probe failure after a successful probe. The default value is 3. The minimum value is 1.

Request method	Description
Command line	With this heath check method, you can detect the container health by executing a probe detection command in the container. The following parameters are supported: · Command: a probe command used to detect the container health. · Initial Delay (in seconds): the initialDelaySeconds parameter, indicating the number of seconds for which the first liveness or readiness probe must wait after the container is started. The default value is 5. · Period (in seconds): the periodseco nds parameter, indicating the interval at which probes are performed. The default value is 10. The minimum value 1. · Timeout (in seconds): the timeoutSeconds parameter, indicating the number of time that the probe has timed out. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes needed for determining a probe success after a failed probe . The default value is 1 and the minimum value is 1. It must be set to 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes needed for determining a probe failure after a successful probe. The default value is 3. The minimum value is 1.

d) Set life cycle rules.

You can set the following parameters for the container life cycle: start, post start, and pre-stop. For more information, see Attach handlers to container lifecycle events.

- Start: Set a pre-start command and parameter for the container.
- Post Start: Set a post-start command for the container.
- Pre Stop: Set a pre-stop command for the container.

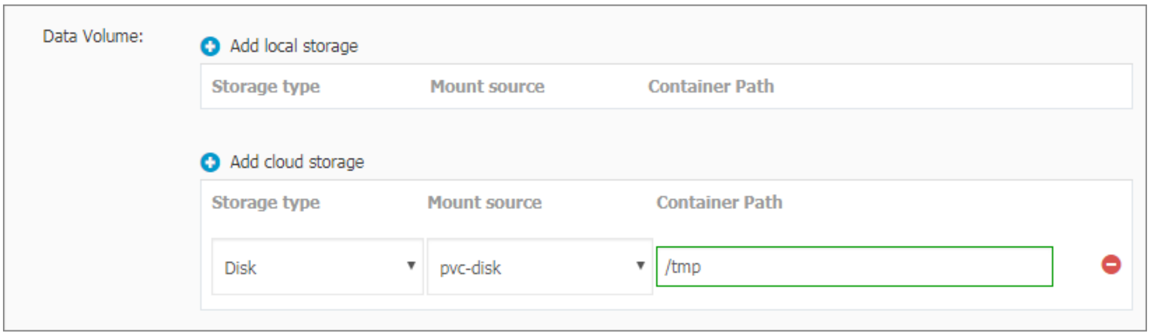


e) Optional: Set volumes.

You can configure local storage and cloud storage.

- local storage: Supported storage types include HostPath, ConfigMap, Secret, and EmptyDir. By setting a type of local storage, you can mount its mount source to the container path. For more information, see Volumes.
- cloud storage: Supported types of cloud storage include disks, Network Attached Storage (NAS), and Object Storage Service (OSS).

This example sets a disk as the volume and mounts the disk to the `/ tmp` container path. Then container data generated in this path is stored to the disk.



f) Optional: Set Log Service. You can set collection parameters and customize tags.

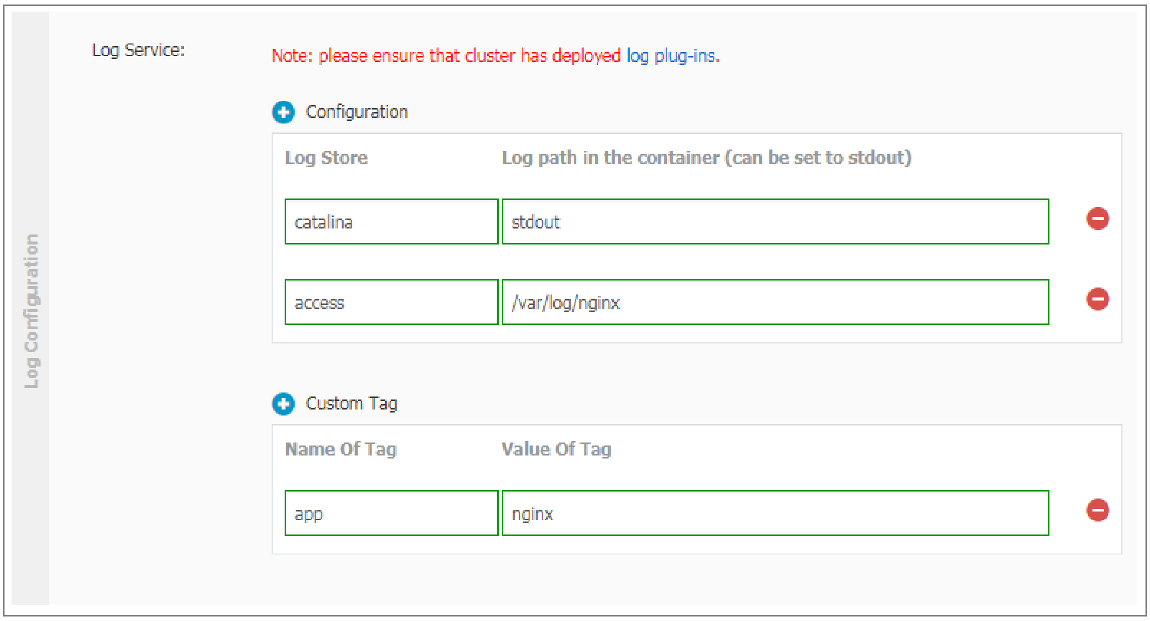
Note:

Make sure that you have deployed a Kubernetes cluster and installed the log plugin on the cluster.

Set the following log collection parameters:

- Log Store: Set a Logstore. After you specify the Logstore name, the Logstore is generated in Log Service to store collected logs.
- Log path in the container: Set this parameter to stdout or set a log path.
  - stdout: If you set the log path parameter to stdout, you can collect the standard output logs of the container.
  - text log: If you specify a container log path, you can collect the text logs of the path. Wildcards can be used in setting the log file name for a log path. In this example, text logs in the path of /var/log/nginx are collected.

You can also customize log tags. The customized log tags can be collected together with container output logs and can benefit log analysis actions such as collecting log statistics and filtering specific logs.



5. Click Next.

6. Configure advanced settings.

   a) Set Access Control.

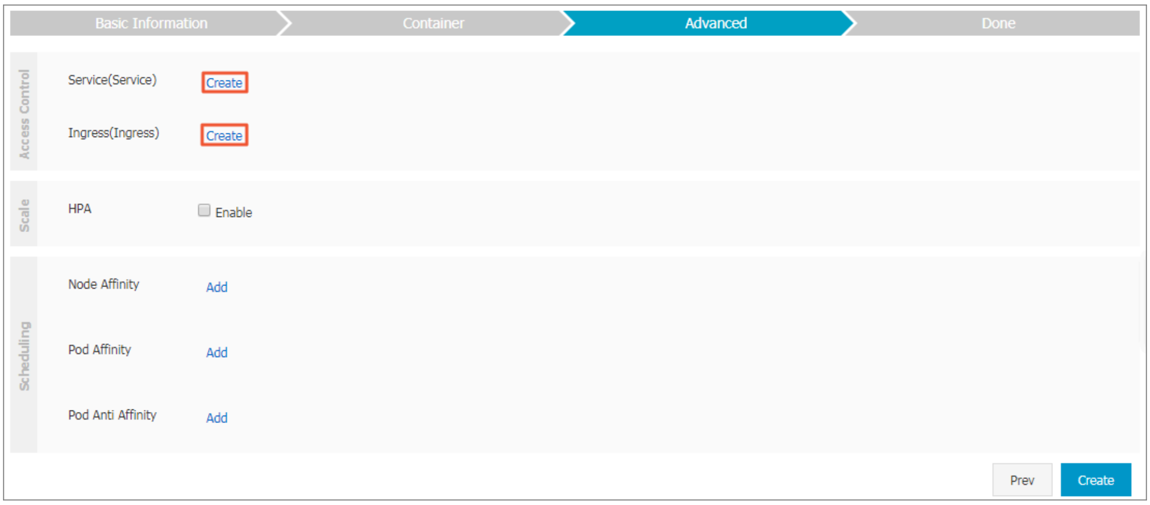
      You can set the methods to expose the application pod and then click Create. In this example, a cluster IP service and an Ingress are set to create an Nginx application that is accessible for the Internet.

      Note:

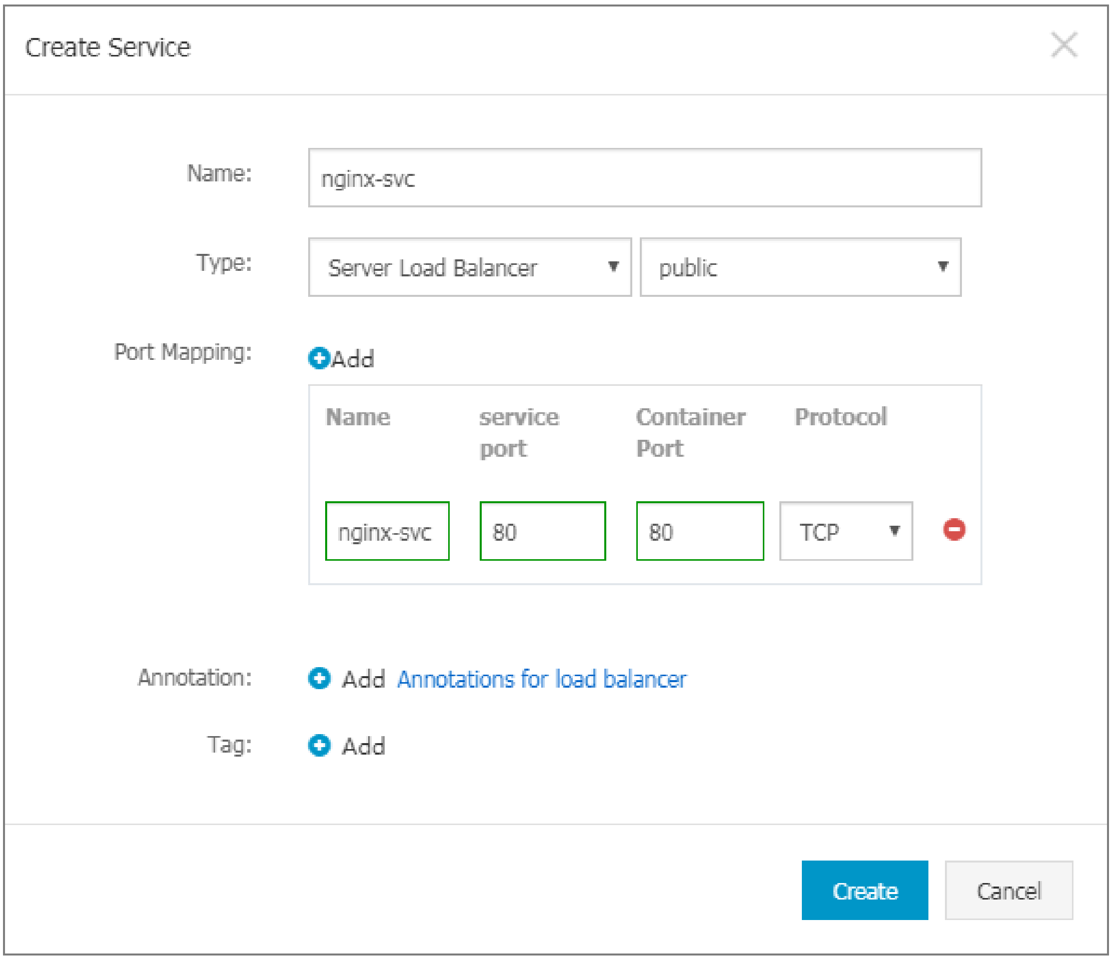
      You can set access methods according to the communication requirements of your application.

      - `Internal applicatio n` : an application that works only inside the cluster. You can create a cluster IP service or a node port service as needed for communication within the cluster.

- `External application` : an application that needs to be exposed to the Internet. You can set how the application is accessed by using either of the following two methods:
  - Create a Server Load Balancer service. This method uses Alibaba Cloud Server Load Balancer (SLB) to provide Internet accessibility for the application.
  - Create a cluster IP service or a node port service, and create an Ingress. This method provides Internet accessibility through the Ingress. For more information, see Ingress.



A. Click Create on the right of Service. Configure a service in the displayed dialog box, and then click Create.



- Name: Enter the service name. The default is `applicatio nname - svc`.
- Type: Select one service type.
  - Cluster IP: Exposes the service by using the internal IP address of your cluster. If you select this service type, the service is accessible only within the cluster.
  - Node port: Exposes the service by using the IP address and the static port (NodePort) of each node. A node port service routes to a cluster IP service that is automatically created. You can access the node port service from outside the cluster by requesting `< NodeIP >:< NodePort >`.
  - Server Load Balancer: Alibaba Cloud Server Load Balancer service. With this type of service, you can set an Internet or intranet access method

for your application. SLB can route to a node port service and a cluster IP service.

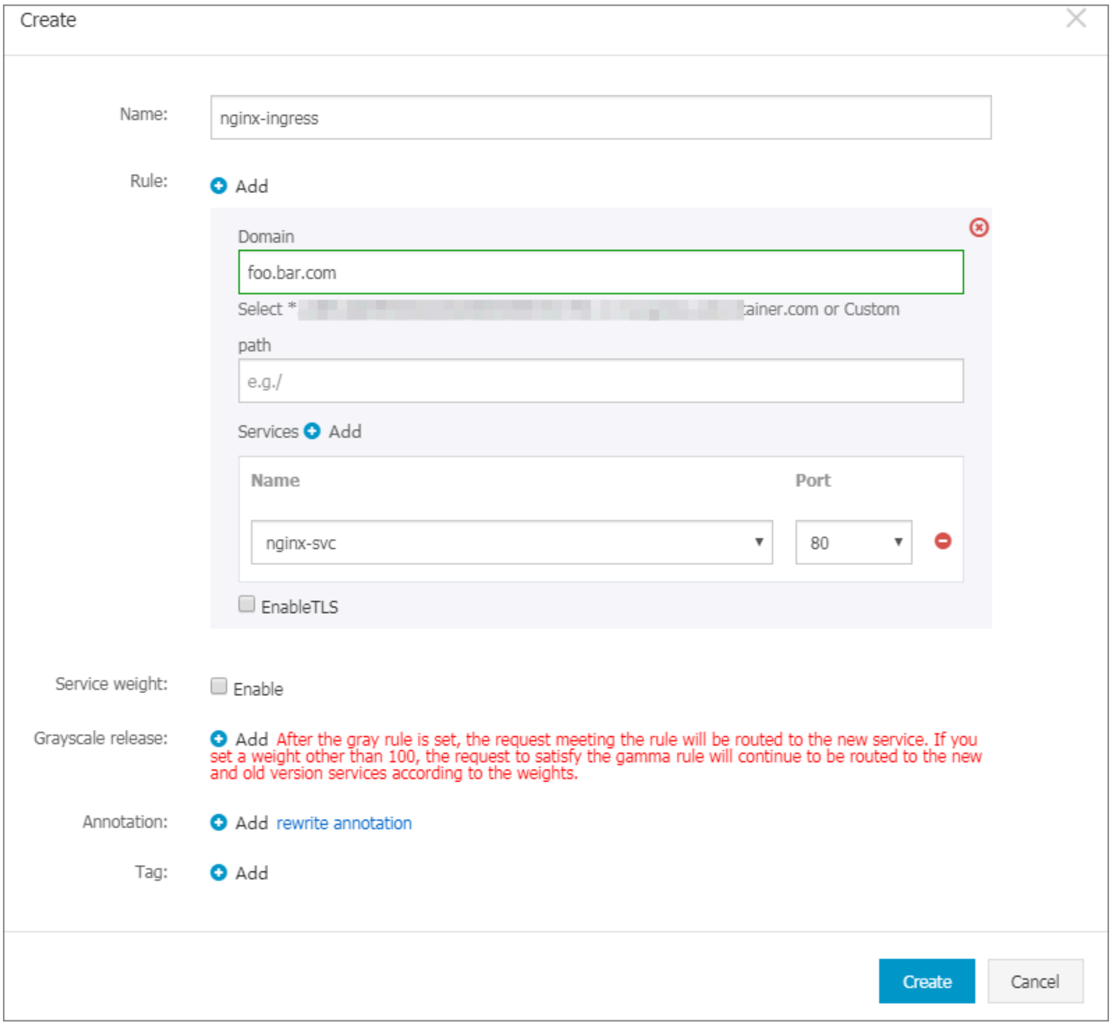
- Port Mapping: Add a service port and a container port, and select the TCP or UDP protocol. If you select the node port Type, you must add a node port to avoid port conflict.
- annotation: Add an annotation to the service. You can set SLB parameters. For more information, see Access services by using Server Load Balancer.
- Tag: Add a tag to the service to identify the service.

B. Click Create on the right of Ingress. In the displayed dialog box, configure an Ingress rule for the application pod, and then click Create. For more information, see Ingress configurations.

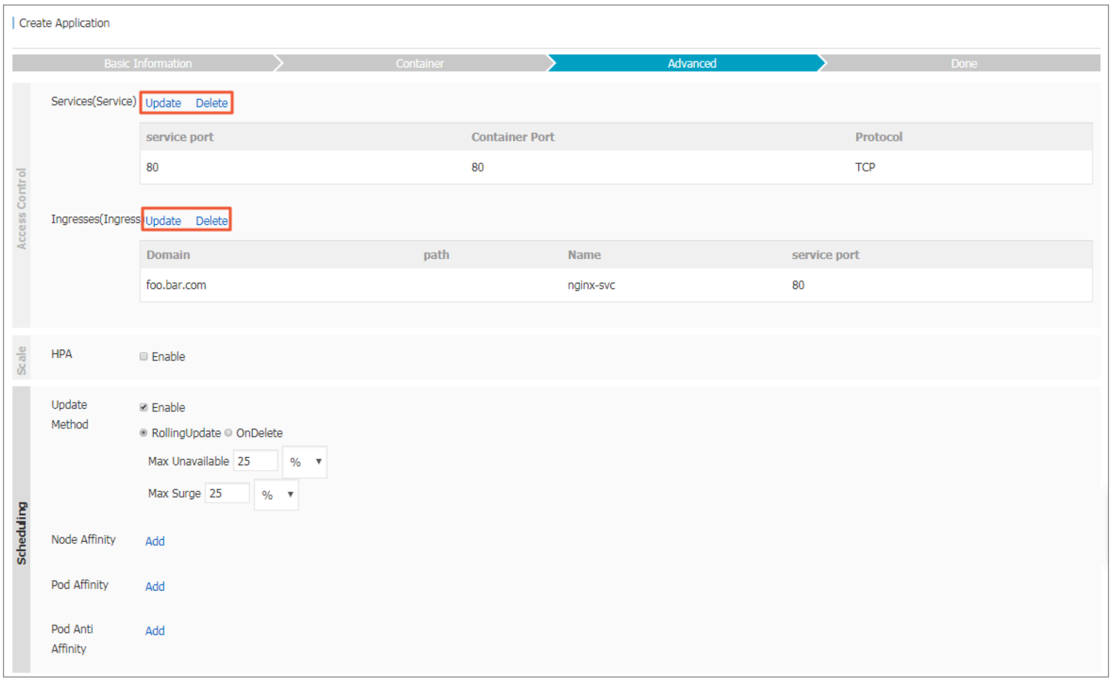
> **Note:**
> When you create an application by using an image, you can create an Ingress rule for only one service. In this example, a virtual host name is used as the

test domain name. You need to add a record to the host. You must use a filing domain name when you create your application.

```
101 . 37 . 224 . 146    foo . bar . com    # This  is  the
  IP  address  of  the  Ingress .
```

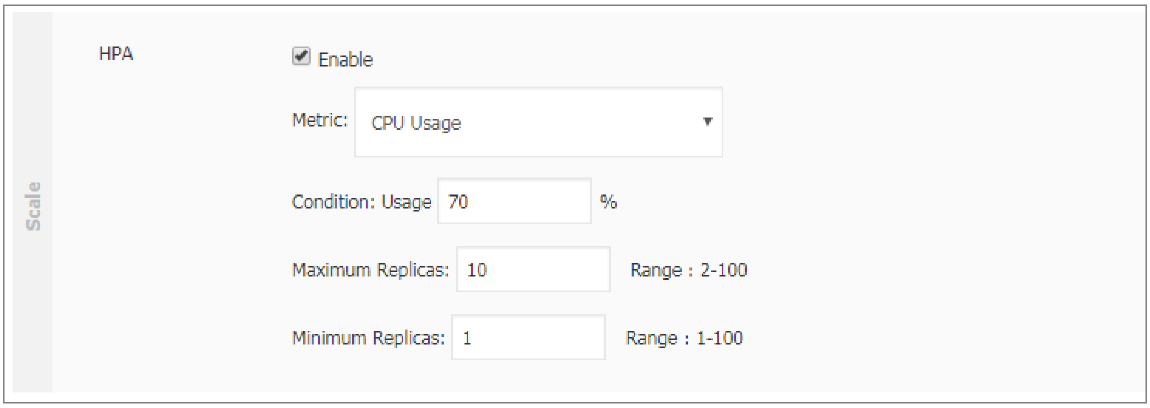


C. In the access control area, the created service and Ingress are displayed. You can perform further configurations by clicking Update or Delete.



b) Optional: Set Horizontal Pod Autoscaling (HPA).

You enable HPA by selecting the Enable check box. Alibaba Cloud Container Service for Kubernetes provides pod auto scaling to deal with different application workloads. That is, you can change the number of pods according to the container CPU and memory usage.



Note:

To use this function, you must set required resources for the pod. Otherwise, pod auto scaling cannot take effect. For more information, see general container settings.

- Metric: resource type. CPU or memory is available. This parameter must be specified with a resource type that is the same as the required resource type.
- Condition: the percentage value of resource usage. The number of containers increases when the resource usage exceeds this value.
- Maximum Replicas: the maximum number of the containers that the deployment can include.
- Minimum Replicas: the minimum number of the containers that the deployment can include.

c) Optional: Set Scheduling.

You can set an update method, node affinity, pod affinity, and pod anti affinity. For more information, see Affinity and anti-affinity.

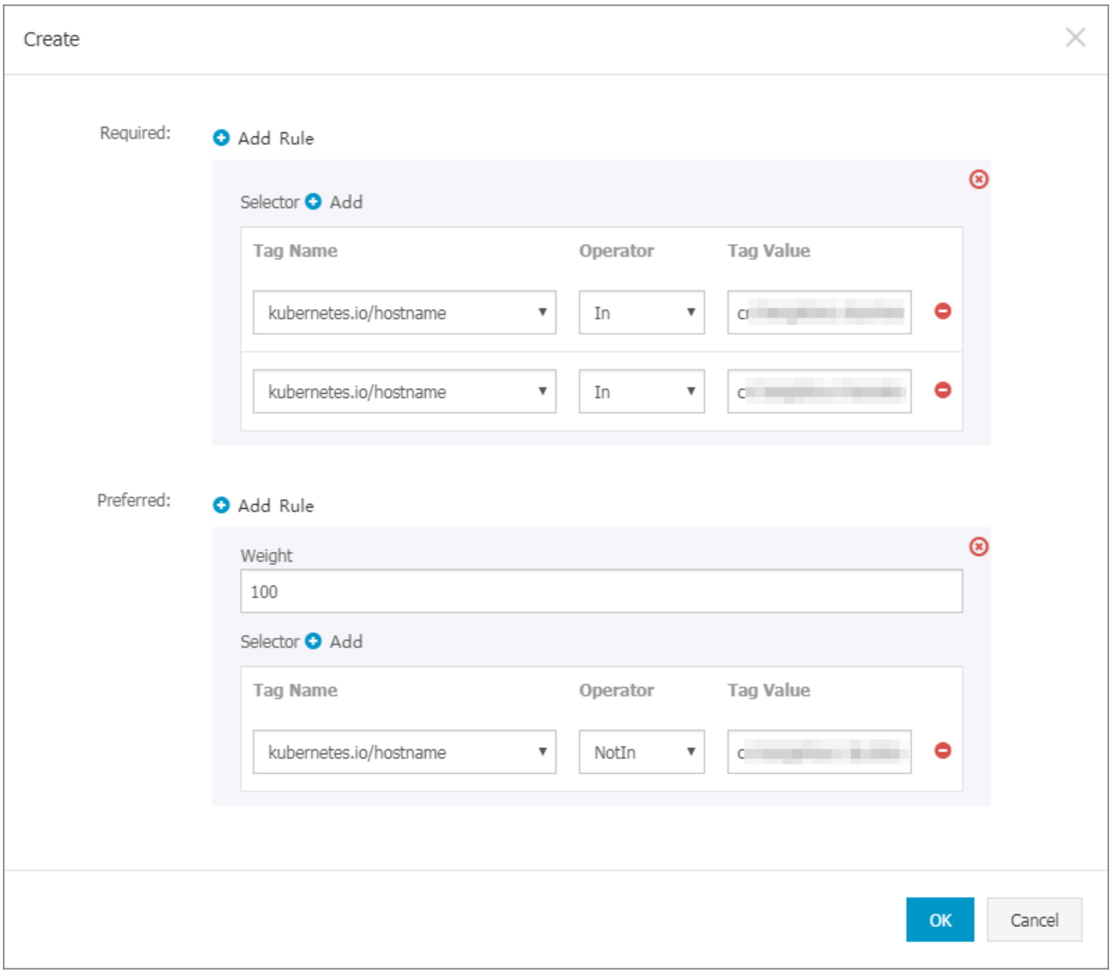
Note:

Affinity scheduling depends on node tags and pod tags. You can use built-in or customized tags to schedule nodes or pods.

A. Set Update Method.

You can select the `RollingUpdate` or `Recreate` (OnDelete) method to replace old pods with new ones. For more information, see Deployments.

B. Set Node Affinity by using node tags.



Required rules and preferred rules are supported, and available operators include `In`, `NotIn`, `Exists`, `DoesNotExist`, `Gt`, and `Lt`.

- Required rules must be satisfied and correspond to `requiredDuringSchedulingIgnoredDuringExecution`. The required rules have the same effect as `NodeSelector`. In this example, the pod can be scheduled to only a node with the specified tags.

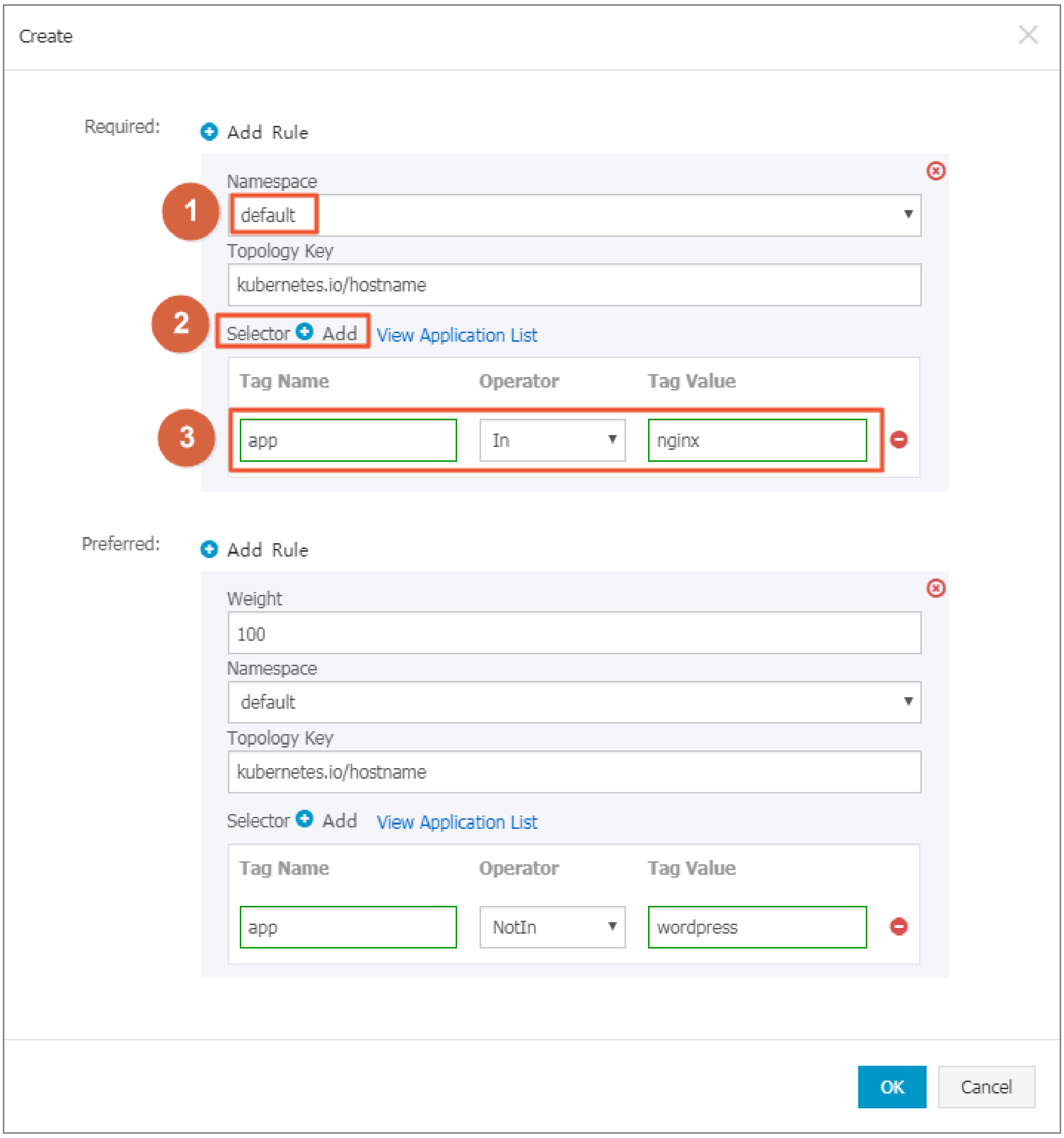
You can add multiple required rules, but only one required rule needs to be satisfied for pod scheduling.

- Preferred rules can be unnecessarily satisfied and correspond to `preferredDuringSchedulingIgnoredDuringExecution`. With the scheduling setting in this example, the system tries not to schedule the pod to the nodes with the specified tag.

  You can also set Weight for each preferred rule. If multiple nodes satisfies the preferred rules, the system schedules the pod to a node with the highest weight.

  You can add multiple preferred rules, and all the rules must be satisfied for pod scheduling.

C. Set Pod Affinity to deploy the application pod in a topology domain together with other pods. For example, to reduce network latency between the services that communicate with each other, you can deploy their pods to a topology domain (for example, a host).



You can schedule pods according to tags of pods running on nodes. Required rules and preferred rules are supported, and available operators include `In`, `NotIn`, `Exists`, `DoesNotExist`.

- Required rules must be satisfied and correspond to `requiredDuringSchedulingIgnoredDuringExecution`. All specified conditions of required rules must be met for pod affinity scheduling.
  - Namespace: Set a namespace. This parameter is required because the scheduling policy is based on pod tags.

- Topology Key: Set a topology domain to which pods are scheduled. This parameter takes effect through node tags. For example, if you set `kubernetes . io / hostname` as the topology key, a node is used to identify a topology. If you set `beta . kubernetes . io / os` as the topology key, a node operating system is used to identify a topology.
- Selector: Click this button to add a required rule.
- View Applicaiton List: Click View Applicaiton List, a dialog box is displayed. In the dialog box, you can view applications in each namespace and export application tags to the dialog box in which you set pod affinity.
- Required rule tag: Set a tag name, its operator, and the tag value for existing applications. This example schedules the application to be created to a host on which applications tagged with `app : nginx` run.

· Preferred rules can be unnecessarily satisfied and correspond to `preferredD uringSched ulingIgnor edDuringEx ecution .` Specified conditions of required rules will be met as many as possible for pod affinity scheduling.

You can set Weight for each preferred rule. The weight value range is 1 to 100. If multiple nodes satisfies the preferred rules, the system schedules the pod to a node with the highest weight. Other parameters are the same with the required rule setting.

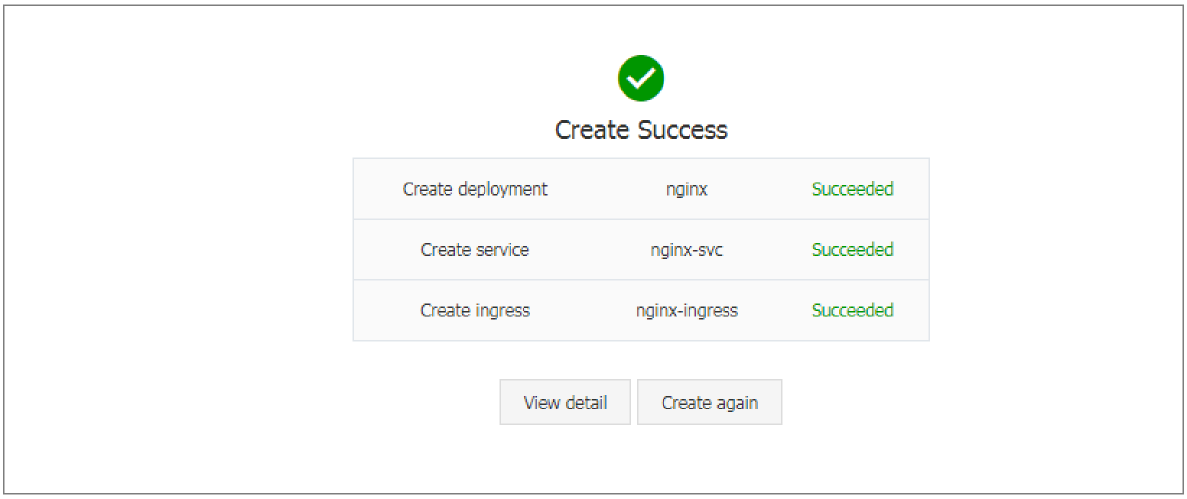
D. Set Pod Anti Affinity to deploy the application pods in a topology domain that excludes other pods. Scenarios that use pod anti affinity scheduling include:

· Distribute the pods of a service to different topology domains (for example , different hosts) to improve the service stability.
· Grant a pod the exclusive access to a node so as to guarantee that no other pods use the resources of the node.
· Distribute pods of the services that may affect each other to different hosts.

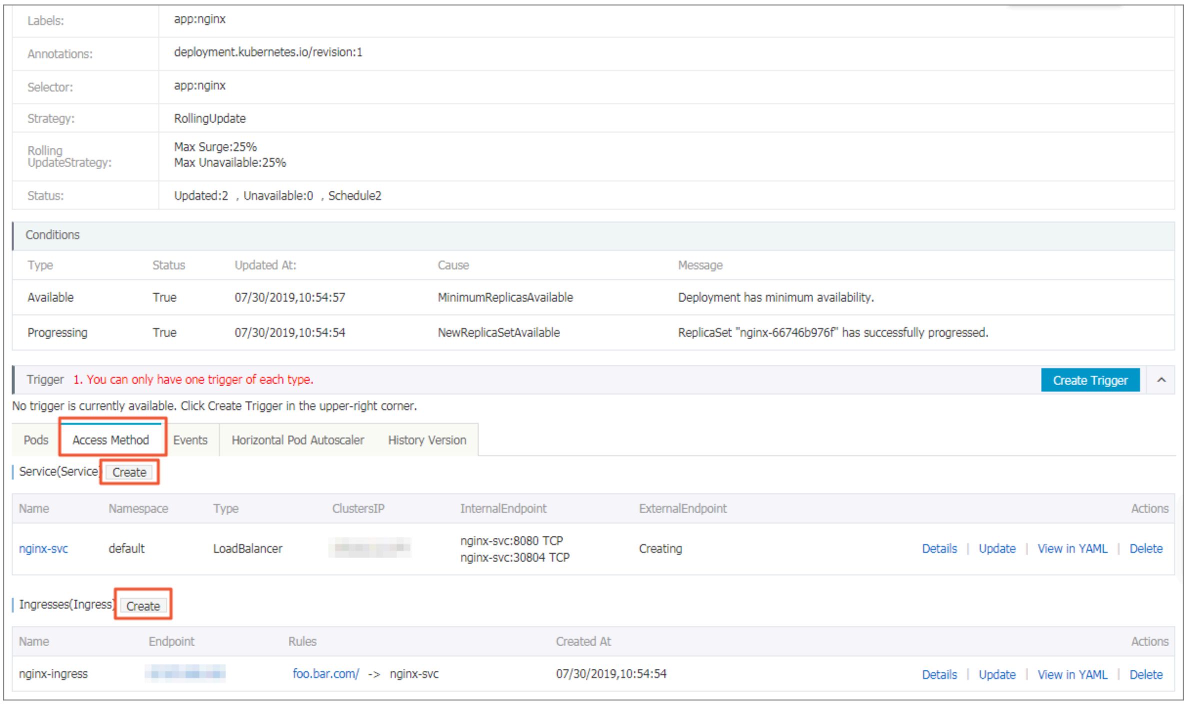
Note:
You can set pod anti affinity scheduling by using the same method as setting pod affinity scheduling. But the same scheduling rules have different

meanings for these two types of scheduling. You need to select appropriate scheduling rules as needed.

7. Click Create.
8. After you create the application, anew page is displayed by default to prompt that you have created the application and lists objects included in the application. You can click View detail to view the deployment details.



The nginx-deployment page is displayed by default.



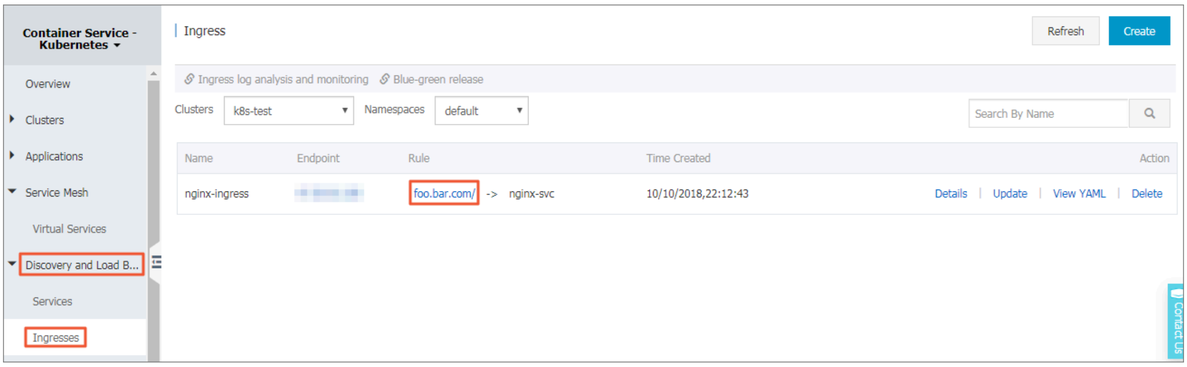
Note:

On the Access Method page, you can also create an Ingress and service as follows:

- On the right of Service, click Create. For more information, see step 6.a.i.

- On the right of Ingress, click Create. For more information, see step 6.a.ii.

9. Choose Discovery and Load Balancing > Ingress to verify that a rule is displayed in the Ingress list.



10. Access the test domain name in your browser to verify that you can visit the Nginx welcome page.

4.2 Create a StatefulSet application by using an image

Kubernetes clusters of Alibaba Cloud Container Service allows you to quickly create applications of the StatefultSet type through the web interface. In this example, create a StatefultSet Nginx application and show features of a StatefultSet application.

Prerequisites

- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have successfully created a cloud disk storage volume claim. For more information, see Create a persistent volume claim.
- You have successfully connected to the master node of the Kubernetes cluster. For more information, see Connect to a Kubernetes cluster by using kubectl.

Context

StatefulSet features are as follows:

Scenarios	Description
Pod consistency	Contains order (such as startup and stop order) and network consistency. This consistency is related to pods and has nothing to do with the node to which the pods are to be scheduled.
Stable persistent storage	Create a PV for each pod through VolumeClaimTemplate. Deleting or reducing replicas does not delete relevant volumes.
Stable network marker	The hostname mode for a pod is: `( statefulse t name )-( sequence number )`.
Stable order	For StatefulSet of N replicas, each pod is assigned a unique order number within the range of 0 to N.

Procedure

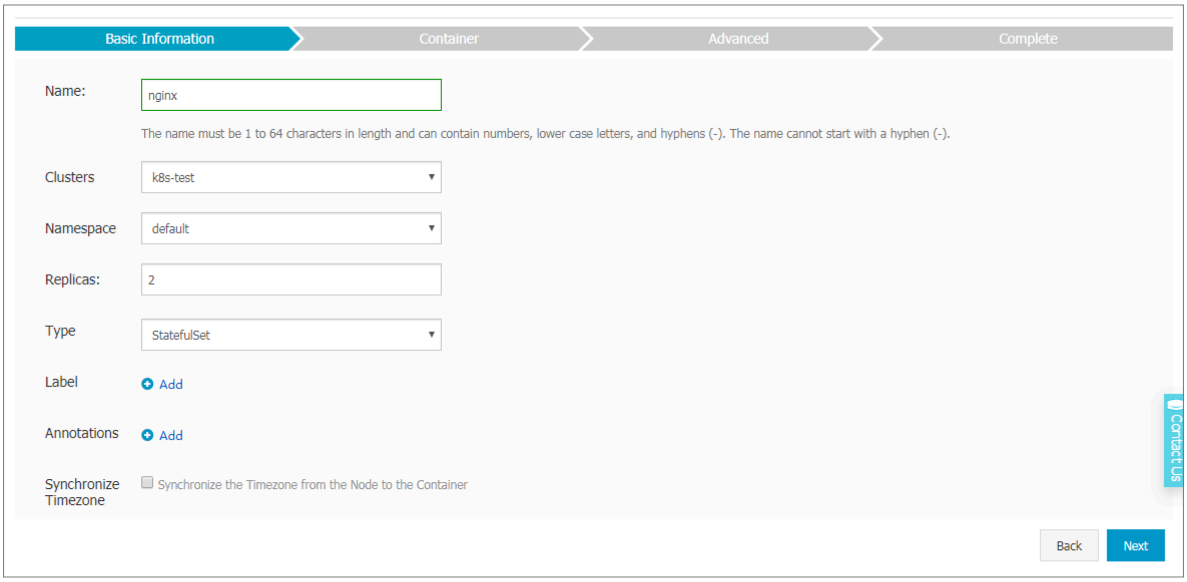
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Application > StatefulSet. Then, click Create from Image in the upper-right corner.
3. Configure the basic parameters and then click Next.

   - Name: Enter the application name.
   - Cluster: Select a cluster to which the application is deployed.
   - Namespace: Select a namespace in which the application deployment is located. By default, the default namespace is used.
   - Replicas: Set the number of pods included in the application.
   - Type: Deployment type and StatefulSet type are available.

   Note:

In this example, select the StatefulSet type.

- Label: Add a label to the application.
- Annotations: Add a annotation to the application.



4. Configure containers.

> **Note:**
> You can configure multiple containers for the pod of the application.

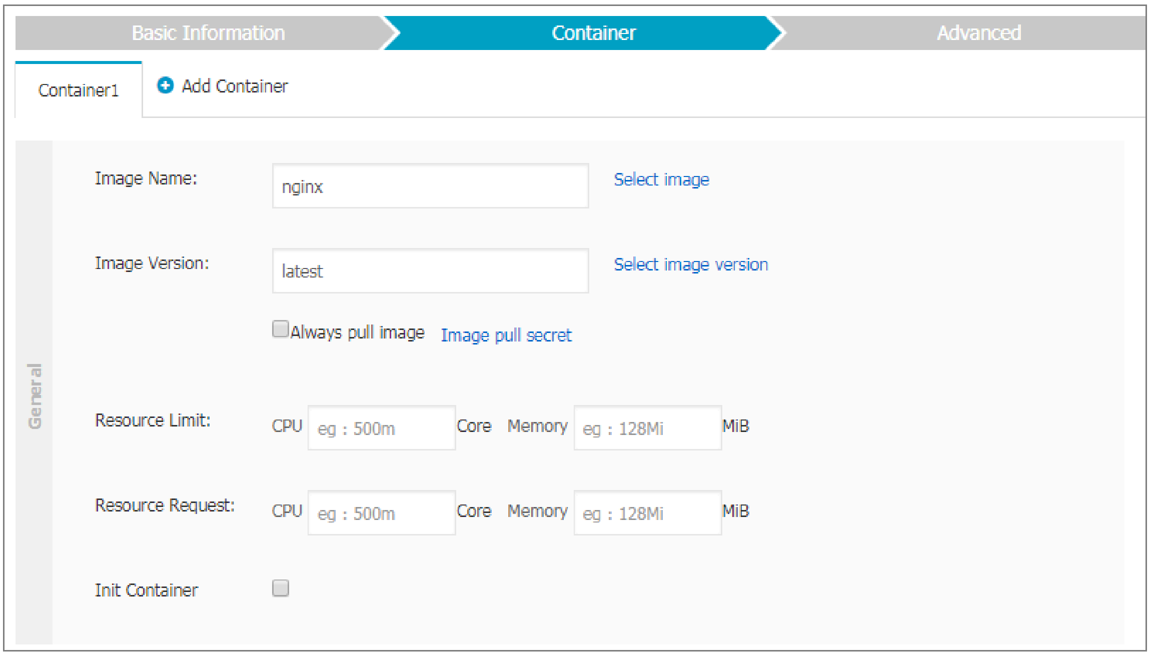
a) Configure the general settings for the application.

- Image Name: Click Select image to select the image in the displayed dialog box and then click OK. In this example, select the nginx image.

  You can also enter the private registry in the format of `domainname / namespace / imagename : tag` to specify an image.
- Image Version: Click Select image version to select a version. If the image version is not specified, the system uses the latest version by default.
- Always pull image: Container Service caches the image to improve deployment efficiency. During deployment, if the image tag is found consistent with that on the local cache, the image on the local cache is reused and is not pulled again. Therefore, if you do not modify the image tag when changing your codes and image for convenience of upper-layer business, the early image on the local cache is used in the application deployment. With this check box selected, Container Service ignores the cached image and re-

pulls the image from the repository when deploying the application to make sure the latest image and codes are always used.

- Resource Limit: Specify the upper limit for the resources (CPU and memory) that can be used by this application to avoid occupying excessive resources. CPU is measured in millicores, that is, one thousandth of one core. Memory is measured in bytes, which can be Gi, Mi, or Ki.
- Resource Request: Specify how many resources (CPU and memory) are reserved for the application, that is, these resources are exclusive to the container. Other services or processes will compete for resources when the resources are insufficient. By specifying the Resource Request, the application will not become unavailable because of insufficient resources.
- Init Container: Selecting this check box creates an Init Container which contains useful tools. For more information, see https://kubernetes.io/docs/concepts/workloads/pods/init-containers/.

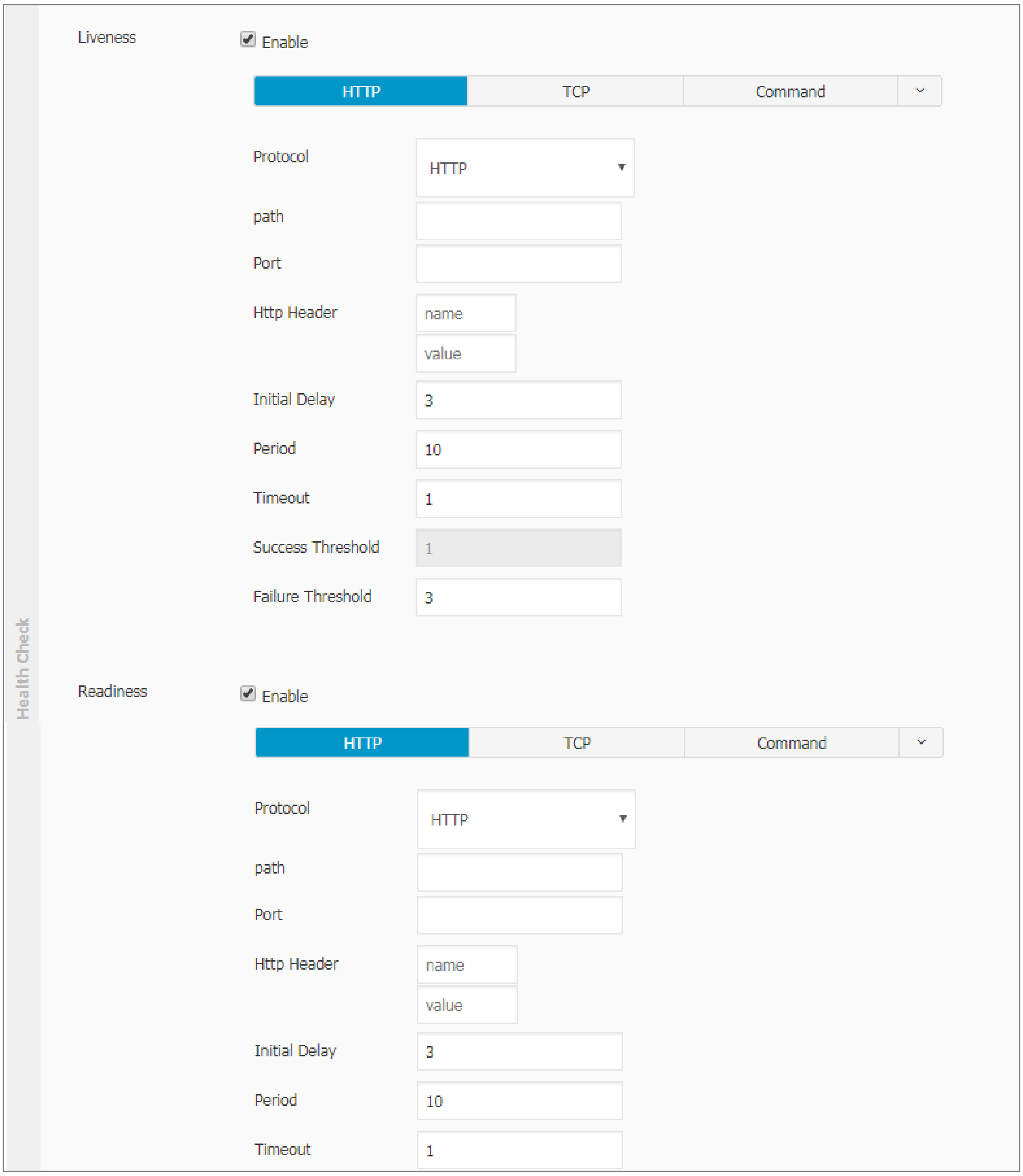


b) Optional: Configure Environment .

You can configure environment variables for the pod by using key-value pairs. Environment variables are used to add environment labels or pass configurations for the pod. For more information, see Pod variable.

c) Optional: Configure Health Check.

The health check function includes liveness probes and readiness probes. Liveness probes are used to detect when to restart the container. Readiness probes determine if the container is ready for receiving traffic. For more information about health check, see https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes.



Request method	Description
HTTP request	An HTTP GET request is sent to the container. The following are supported parameters: · Protocol: HTTP/HTTPS · Path: Path to access the HTTP server · Port: Number or name of the port exposed by the container. The port number must be in the range of 1 to 65535. · HTTP Header: Custom headers in the HTTP request. HTTP allows repeated headers. Supports the correct configuration of key values. · Initial Delay (in seconds): Namely, the initialDelaySeconds. Seconds for the first probe has to wait after the container is started. The default is 3. · Period (in seconds): Namely, the periodseconds. Intervals at which the probe is performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): Namely, the timeoutSeconds. The time of probe timeout. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes that are considered as successful after a failed probe. The default is 1 and the minimum is 1. It must be 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes that are considered as failed after a successful probe. The default value is 3. The minimum value is 1.

Request method	Description
TCP connection	A TCP socket is send to the container. The kubelet attempts to open a socket to your container on the specified port. If a connection can be established, the container is considered healthy. If not, it is considered as a failure. The following are supported parameters: · Port: Number or name of the port exposed by the container. The port number must be in the range of 1 to 65535. · Initial Delay (in seconds): Namely, the initialDelaySeconds. Seconds for the first liveness or readiness probe has to wait after the container is started. The default is 15. · Period (in seconds): Namely, the periodseconds. Intervals at which the probe is performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): Namely, the timeoutSeconds. The time of probe timeout. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes that are considered as successful after a failed probe. The default is 1 and the minimum is 1. It must be 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes that are considered as failed after a successful probe. The default value is 3. The minimum value is 1.

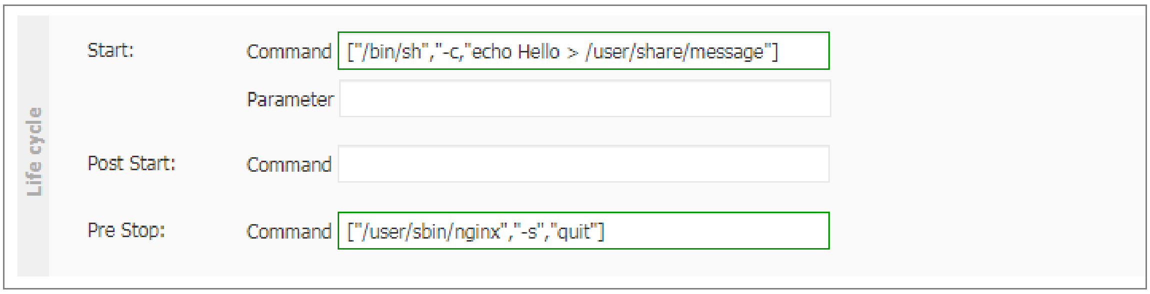
Request method	Description
Command line	Detect the health of the container by executing probe detection commands in the container. The following are supported parameters: · Command: A probe command used to detect the health of the container . · Initial Delay (in seconds): Namely, the initialDelaySeconds. Seconds for the first liveness or readiness probe has to wait after the container is started. The default is 5. · Period (in seconds): Namely, the periodseconds. Intervals at which the probe is performed. The default value is 10. The minimum value 1. · Timeout (in seconds): Namely, the timeoutSeconds. The time of probe timeout. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes that are considered as successful after a failed probe. The default is 1 and the minimum is 1. It must be 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes that are considered as failed after a successful probe. The default value is 3. The minimum value is 1.

d) Optional: Configure the lifecycle rule.

You can configure the following parameters for the container lifecycle: start, post start, and pre-stop. For more information, see https://kubernetes.io/docs/tasks/configure-pod-container/attach-handler-lifecycle-event/.

- Start: Configure a pre-start command and parameter for the container.
- Post Start: Configure a post-start command for the container.

- Pre Stop: Configure a pre-end command for the container.

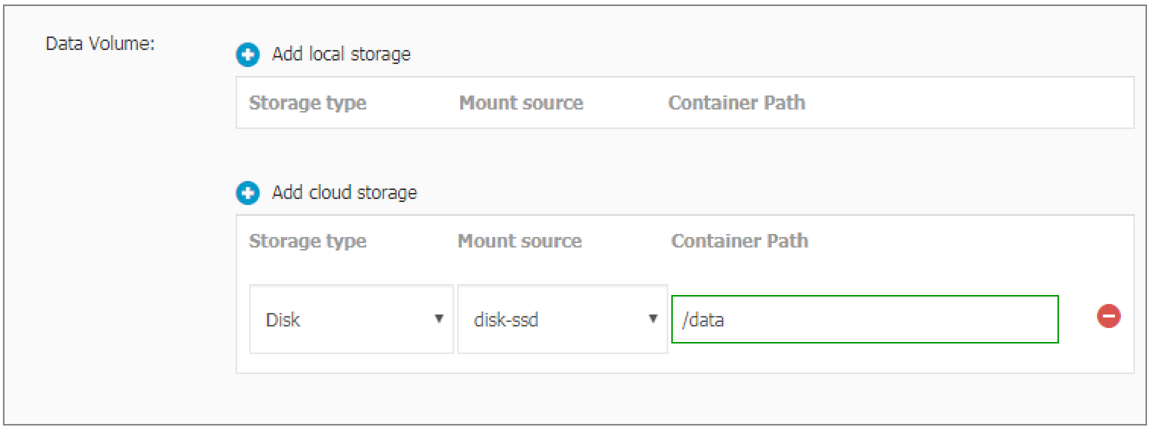


e) Configure data volumes.

Local storage and cloud storage can be configured.

- Local storage: Supports hostPath, configmap, secret, and temporary directory. The local data volumes mount the corresponding mount source to the container path. For more information, see Volumes.
- Cloud storage: Supports three types of cloud storage: cloud disks, Network Attached Storage (NAS), and Object Storage Service (OSS).

In this example, configure a data volume claim named disk-ssd of cloud disk type and mount it to the `/ data` path.



f) Optional: Configure Log Service. You can configure collection methods and customize tags for this service.

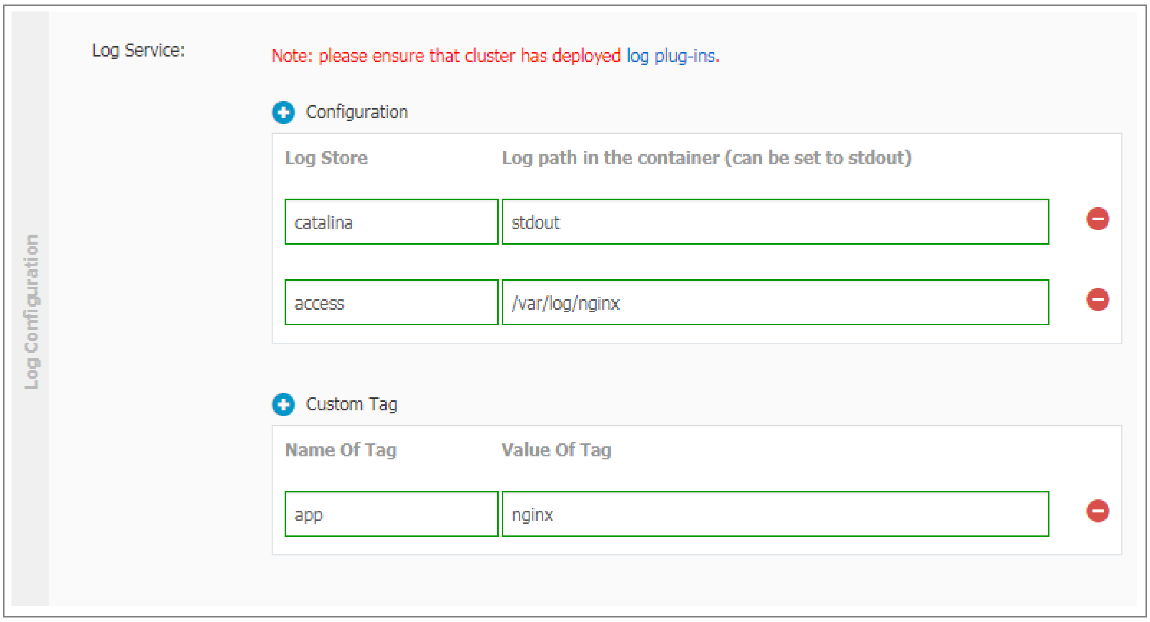
Note:

Make sure that a Kubernetes cluster is deployed and that the log plug-in is installed on the cluster.

Configure log collection methods as follows:

- Log Store: Configure a Logstore generated in Log Service which is used to store collected logs.
- Log path in the container: Supports stdout and text logs.
  - stdout: Collects standard output logs of containers.
  - text log: Collects logs in the specified path in the container. In this example, collect text logs in the path of /var/log/nginx. Wildcards are also supported.

You can also set custom tags. The customized tags are collected to the container output logs. A custom tag can help you tag container logs, providing convenienc e to log analysis such as log statistics and filter.



5. Click Next after completing the configurations.
6. Configure advanced settings. In this example, configure only access settings.
   a) Set Access Control.

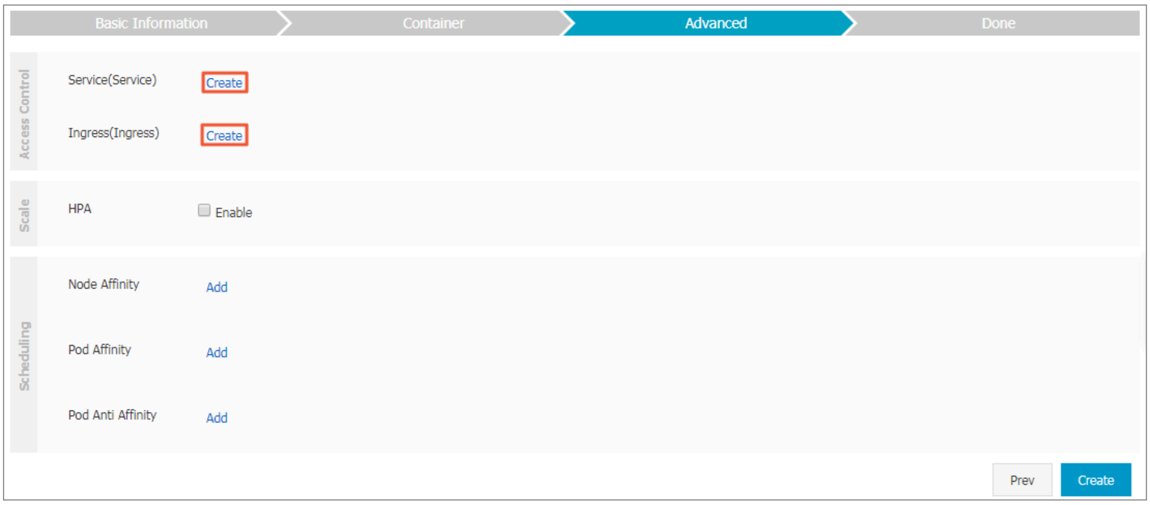
      You can set the methods to expose the application pod and then click Create. In this example, a cluster IP service and an Ingress are set to create an Nginx application that is accessible for the Internet.

      Note:

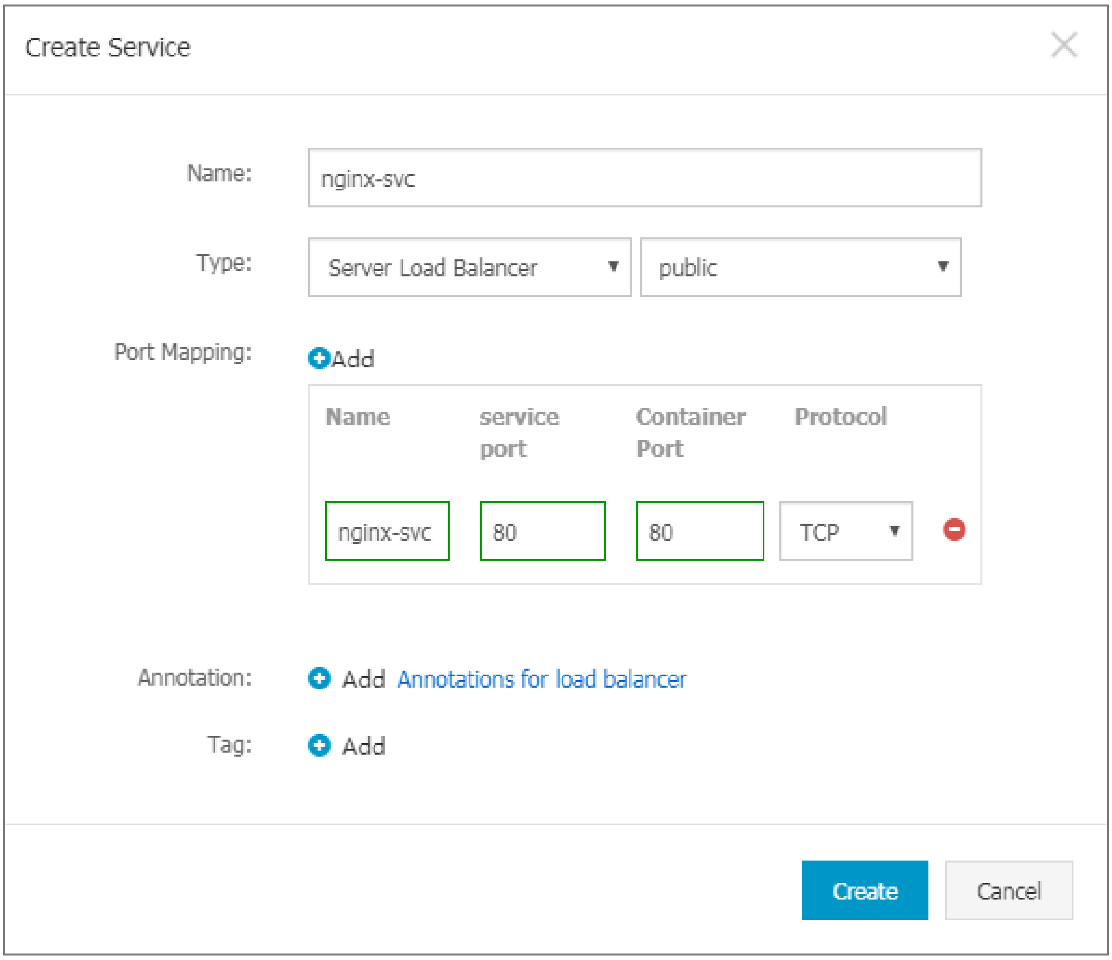
You can set access methods according to the communication requirements of your application.

- `Internal application` : an application that works only inside the cluster. You can create a cluster IP service or a node port service as needed for communication within the cluster.

- `External application`: an application that needs to be exposed to the Internet. You can set how the application is accessed by using either of the following two methods:
  - Create a Server Load Balancer service. This method uses Alibaba Cloud Server Load Balancer (SLB) to provide Internet accessibility for the application.
  - Create a cluster IP service or a node port service, and create an Ingress. This method provides Internet accessibility through the Ingress. For more information, see Ingress.



A. Click Create on the right of Service. Configure a service in the displayed dialog box, and then click Create.



- Name: Enter the service name. The default is `applicatio nname - svc`.
- Type: Select one service type.
  - Cluster IP: Exposes the service by using the internal IP address of your cluster. If you select this service type, the service is accessible only within the cluster.
  - Node port: Exposes the service by using the IP address and the static port (NodePort) of each node. A node port service routes to a cluster IP service that is automatically created. You can access the node port service from outside the cluster by requesting `< NodeIP >:< NodePort >`.
  - Server Load Balancer: Alibaba Cloud Server Load Balancer service. With this type of service, you can set an Internet or intranet access method

for your application. SLB can route to a node port service and a cluster IP service.

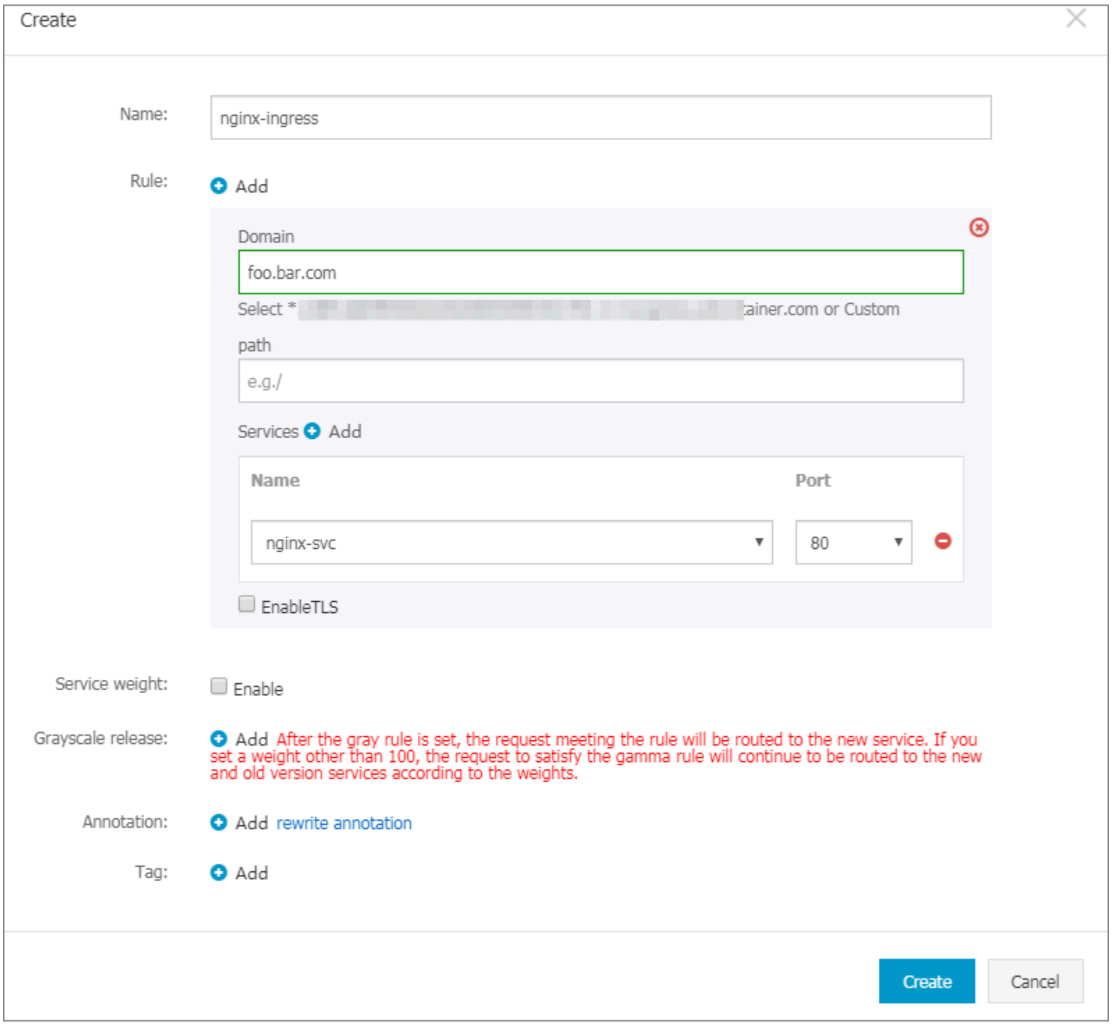
- Port Mapping: Add a service port and a container port, and select the TCP or UDP protocol. If you select the node port Type, you must add a node port to avoid port conflict.
- annotation: Add an annotation to the service. You can set SLB parameters. For more information, see Access services by using Server Load Balancer.
- Tag: Add a tag to the service to identify the service.

B. Click Create on the right of Ingress. In the displayed dialog box, configure an Ingress rule for the application pod, and then click Create. For more information, see Ingress configurations.

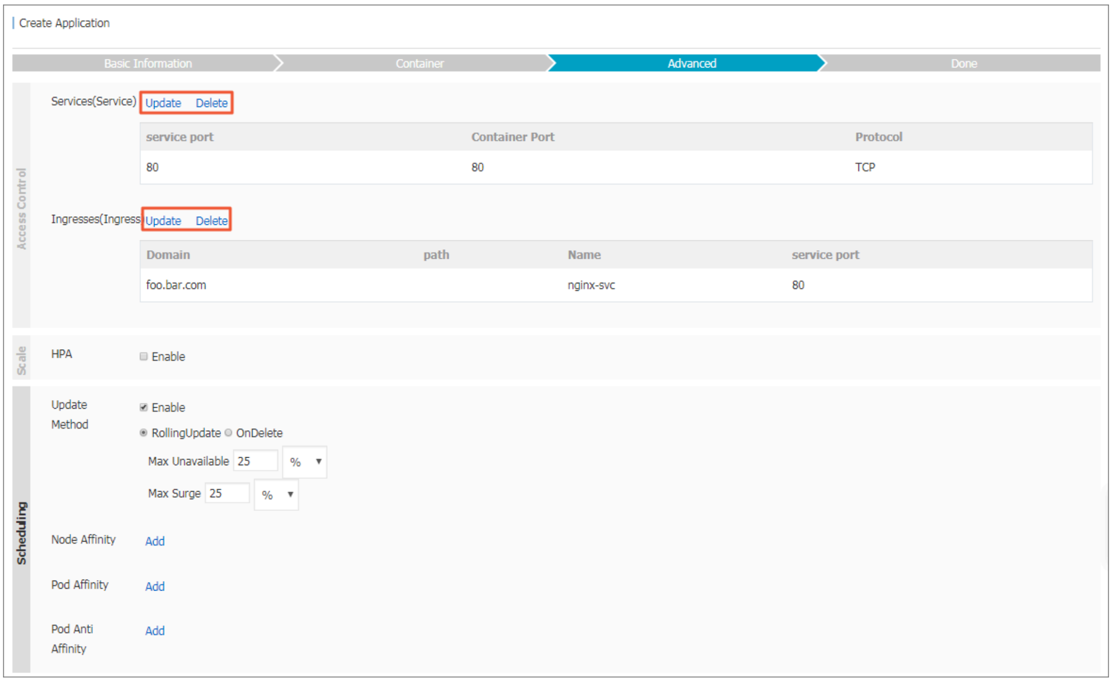
> Note:
> When you create an application by using an image, you can create an Ingress rule for only one service. In this example, a virtual host name is used as the

test domain name. You need to add a record to the host. You must use a filing domain name when you create your application.

```
101 . 37 . 224 . 146     foo . bar . com     # This   is   the
  IP   address   of   the   Ingress .
```

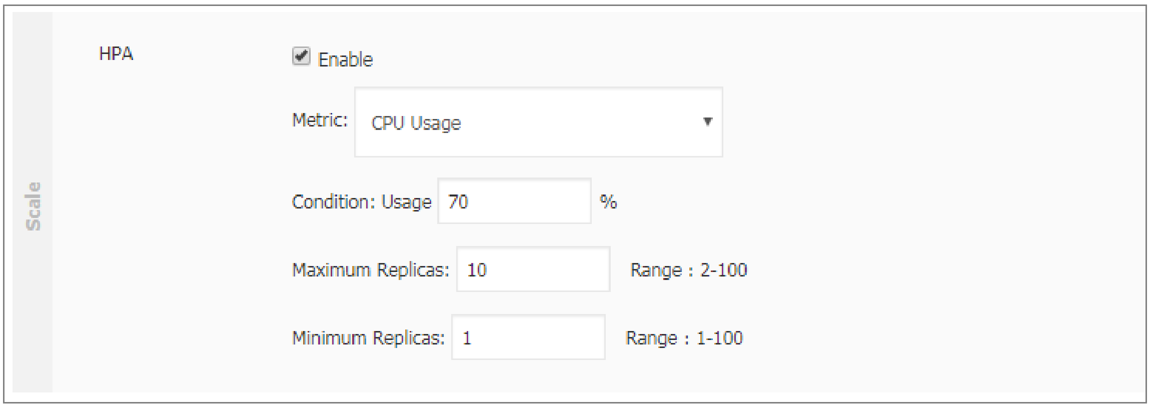


C. In the access control area, the created service and Ingress are displayed. You can perform further configurations by clicking Update or Delete.



b) Optional: Set Horizontal Pod Autoscaling (HPA).

You enable HPA by selecting the Enable check box. Alibaba Cloud Container Service for Kubernetes provides pod auto scaling to deal with different application workloads. That is, you can change the number of pods according to the container CPU and memory usage.



Note:

To use this function, you must set required resources for the pod. Otherwise, pod auto scaling cannot take effect. For more information, see general container settings.

- Metric: resource type. CPU or memory is available. This parameter must be specified with a resource type that is the same as the required resource type.
- Condition: the percentage value of resource usage. The number of containers increases when the resource usage exceeds this value.
- Maximum Replicas: the maximum number of the containers that the StatefulSet application can contain.
- Minimum Replicas: the minimum number of the containers that the StatefulSet application can contain.

c) Optional: Set Scheduling.

You can set an update method, node affinity, pod affinity, and pod anti affinity. For more information, see Affinity and anti-affinity.

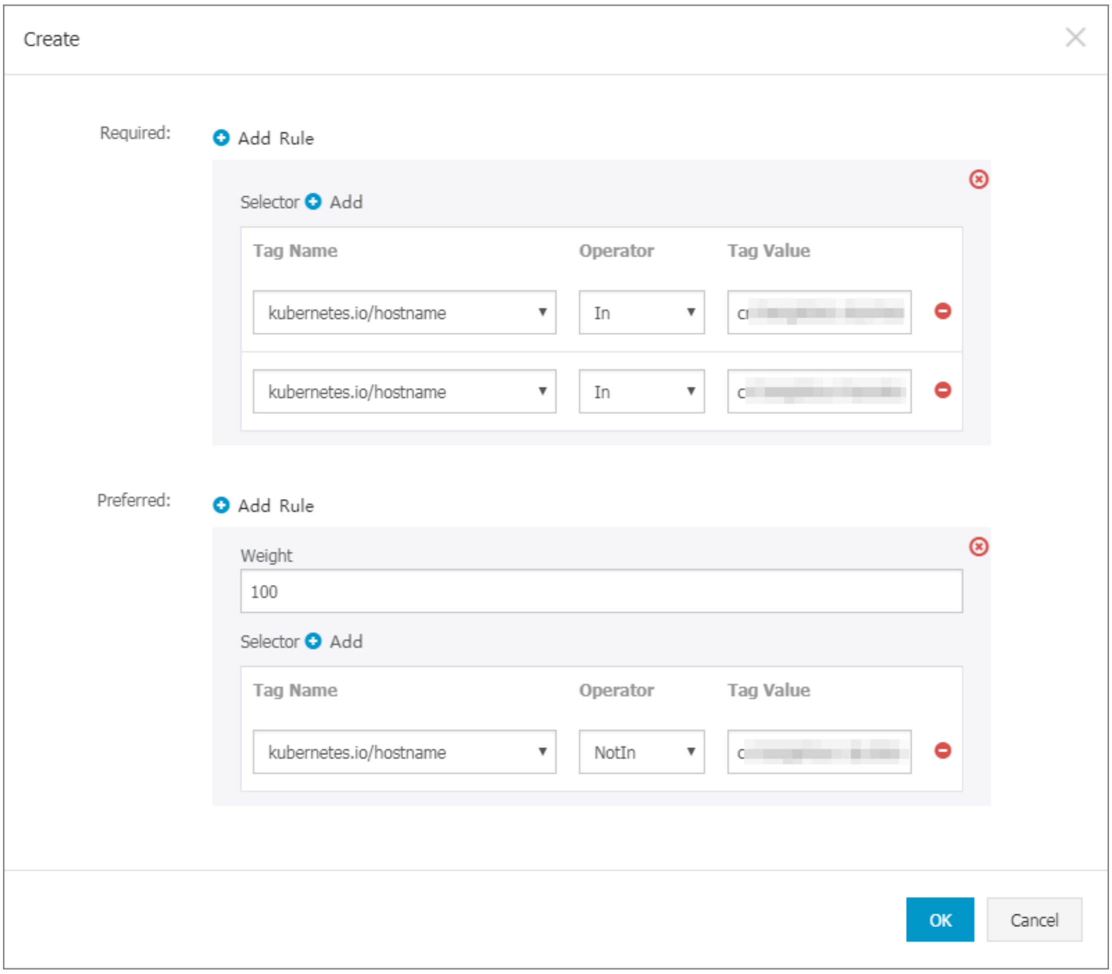
Note:

Affinity scheduling depends on node tags and pod tags. You can use built-in or customized tags to schedule nodes or pods.

A. Set Update Method.

You can select the `RollingUpd ate` or `Recreate` (OnDelete) method to replace old pods with new ones. For more information, see Deployments.

B. Set Node Affinity by using node tags.



Required rules and preferred rules are supported, and available operators include `In`, `NotIn`, `Exists`, `DoesNotExi st`, `Gt`, and `Lt`.

- Required rules must be satisfied and correspond to `requiredDu ringSchedu lingIgnore dDuringExe cution`. The required rules have the same effect as `NodeSelect or`. In this example, the pod can be scheduled to only a node with the specified tags.

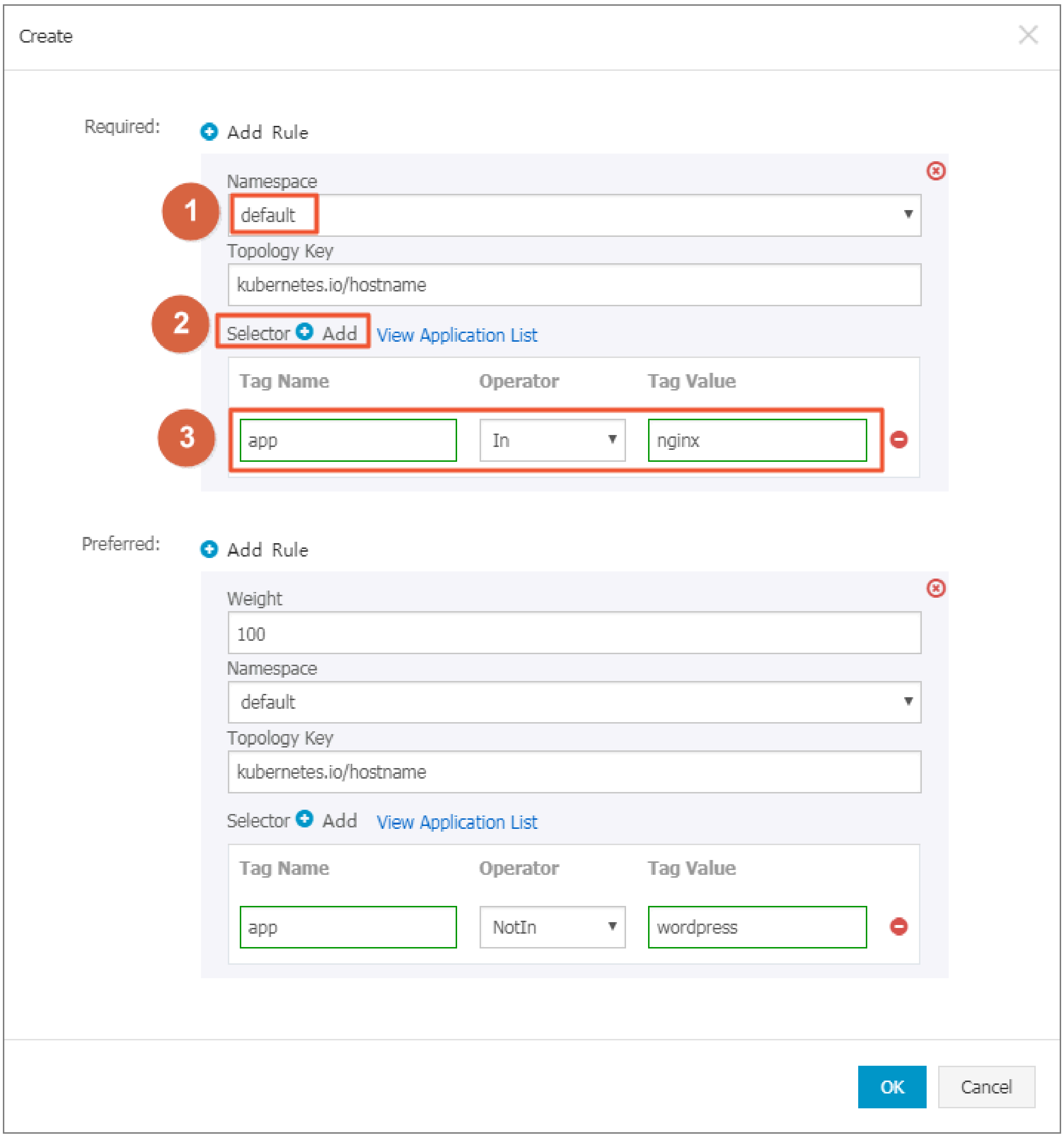
You can add multiple required rules, but only one required rule needs to be satisfied for pod scheduling.

- Preferred rules can be unnecessarily satisfied and correspond to `preferredD uringSched ulingIgnor edDuringEx ecution`. With the scheduling setting in this example, the system tries not to schedule the pod to the nodes with the specified tag.

  You can also set Weight for each preferred rule. If multiple nodes satisfies the preferred rules, the system schedules the pod to a node with the highest weight.

  You can add multiple preferred rules, and all the rules must be satisfied for pod scheduling.

C. Set Pod Affinity to deploy the application pod in a topology domain together with other pods. For example, to reduce network latency between the services that communicate with each other, you can deploy their pods to a topology domain (for example, a host).



You can schedule pods according to tags of pods running on nodes. Required rules and preferred rules are supported, and available operators include `In`, `NotIn`, `Exists`, `DoesNotExist`.

- Required rules must be satisfied and correspond to `requiredDuringSchedulingIgnoredDuringExecution`. All specified conditions of required rules must be met for pod affinity scheduling.
  - Namespace: Set a namespace. This parameter is required because the scheduling policy is based on pod tags.

- Topology Key: Set a topology domain to which pods are scheduled. This parameter takes effect through node tags. For example, if you set `kubernetes . io / hostname` as the topology key, a node is used to identify a topology. If you set `beta . kubernetes . io / os` as the topology key, a node operating system is used to identify a topology.
- Selector: Click this button to add a required rule.
- View Applicaiton List: Click View Applicaiton List, a dialog box is displayed. In the dialog box, you can view applications in each namespace and export application tags to the dialog box in which you set pod affinity.
- Required rule tag: Set a tag name, its operator, and the tag value for existing applications. This example schedules the application to be created to a host on which applications tagged with `app : nginx` run.

· Preferred rules can be unnecessarily satisfied and correspond to `preferredD uringSched ulingIgnor edDuringEx ecution .` Specified conditions of required rules will be met as many as possible for pod affinity scheduling.

You can set Weight for each preferred rule. The weight value range is 1 to 100. If multiple nodes satisfies the preferred rules, the system schedules the pod to a node with the highest weight. Other parameters are the same with the required rule setting.

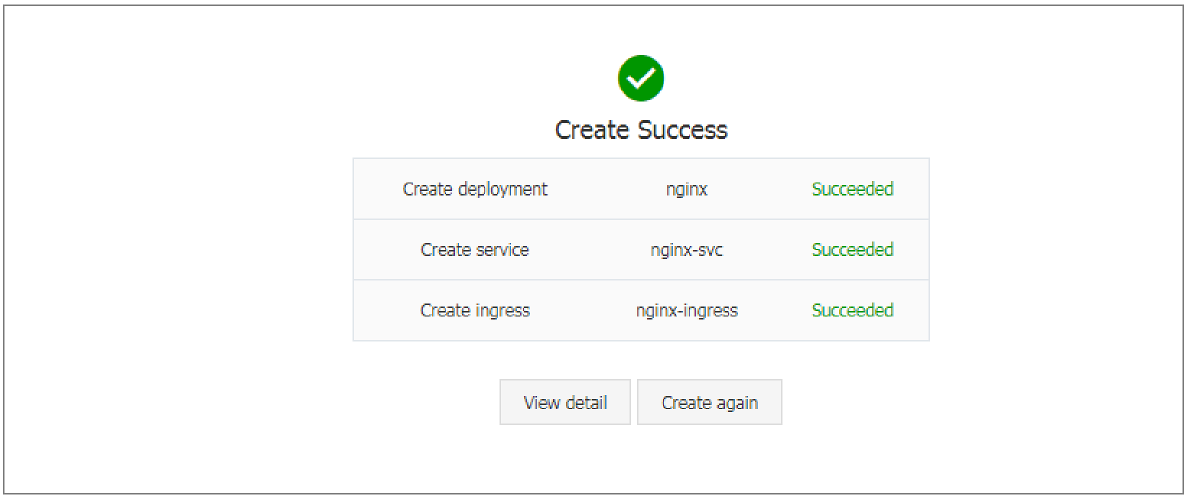
D. Set Pod Anti Affinity to deploy the application pods in a topology domain that excludes other pods. Scenarios that use pod anti affinity scheduling include:

· Distribute the pods of a service to different topology domains (for example , different hosts) to improve the service stability.
· Grant a pod the exclusive access to a node so as to guarantee that no other pods use the resources of the node.
· Distribute pods of the services that may affect each other to different hosts.

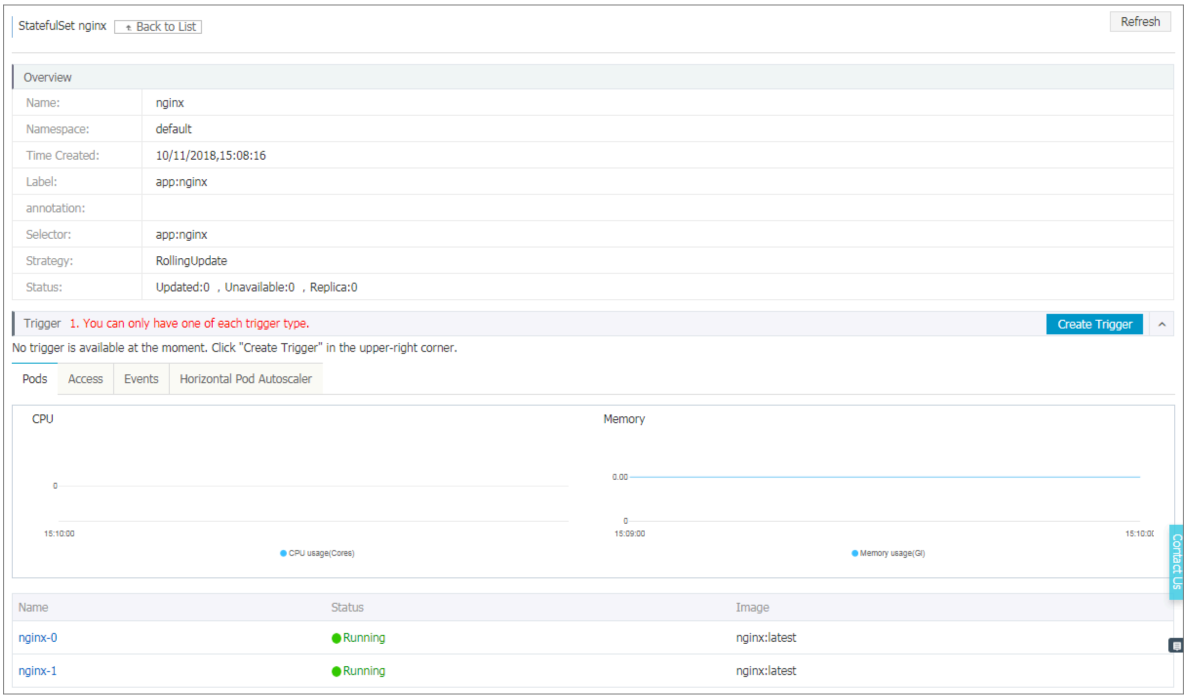
Note:
You can set pod anti affinity scheduling by using the same method as setting pod affinity scheduling. But the same scheduling rules have different

> meanings for these two types of scheduling. You need to select appropriate scheduling rules as needed.

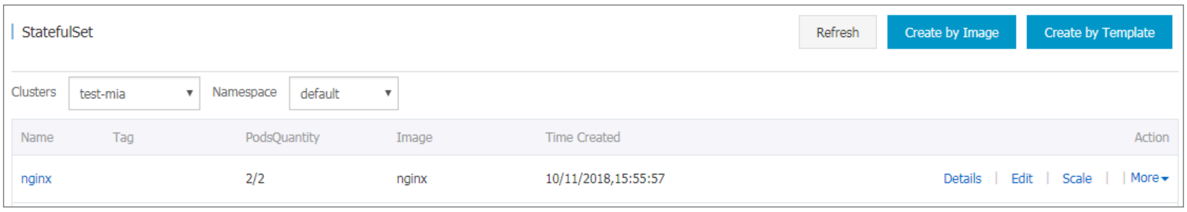
7. Click Create.
8. After you create the application, the create success page is displayed by default and objects contained in the application are listed. You can click View detail to view the deployment details.



The StatefulSet page is displayed by default.

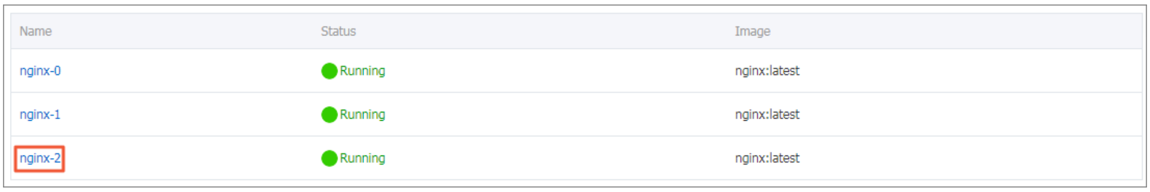


9. Then click Back to list in the upper-left corner to view the created StatefulSet application in the StatefulSet list page.

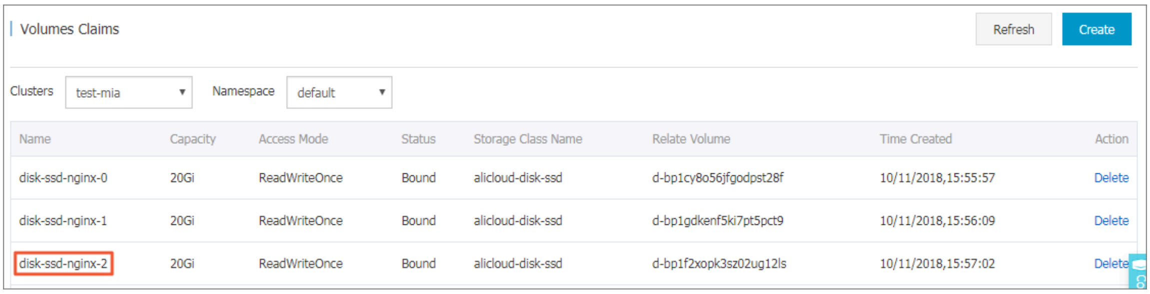


10. Optional: To verify service scalability, click Scale at the right of a target nginx application.

    a) In the displayed dialog box, set the number of pod to 3. You can see that when you expand pods, the pods are in the increment order; when you contract pods, the pods are in the descending order. This shows the order stability of pods in StatefulSet.



    b) Click Application > Volumes Claim in the left-side navigation pane, you can see that as the application expands, new cloud disk volumes are created with pods; if the application contracts, created PV/PVC will not be deleted.



What's next

Connect to the master node and run following commands to verify the persistent storage feature.

Create a temporary file on a cloud disk:

```
#  kubectl   exec   nginx - 1   ls  / tmp              # list   files
 under   this   directory
 lost + found

#  kubectl   exec   nginx - 1   touch   / tmp / statefulse  t
 # add   a   temporty   file   named   statefulse  t
```

```
# kubectl exec nginx - 1 ls / tmp
 lost + found
 statefulse t
```

Remove the pod to verify the data persistence:

```
# kubectl delete pod nginx - 1
 pod " nginx - 1 " deleted

# kubectl exec nginx - 1 ls / tmp                          #
 data persistenc e storage
 lost + found
 statefulse t
```

In addition, you can also find that after you delete a pod, the pod automatically restarts after a period of time, which indicates the high availability of the StatefulSet application.

4.3 Create a Job application by using an image

By running a Kubernetes cluster with Alibaba Cloud Container Service, you can create a Job application through the Web interface. This example creates a Job application named busybox to describe features of the Job application features.

Prerequisites

You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.

Context

A Job processes short-lived one-off tasks in batches to guarantee that one or multiple pods in the batch tasks successfully terminate.

Kubernetes supports the following types of Jobs:

- Non-parallel Job: A Job of this type creates only one pod. The Job is completed when the pod terminates successfully.
- Job with a fixed completion count: A Job of this type has `. spec . completion s` set to create multiple pods. The Job is completed when the number of these pods reaches the `. spec . completion s` value.
- Parallel Job with a work queue: A Job of this type has `. spec . Parallelis m` set but has `. spec . completion s` not set. The Job is completed when at least one pod has terminated with success, and all pods are terminated.

- Parallel Job with a fixed completion count: A Job of this type has both `.spec.completions` and `.spec.Parallelism` set. Multiple pods of the Job process the work queue at the same time.

According to the `.spec.completions` and `.spec.Parallelism` settings, Jobs can be classified into the following patterns.

Note:
The Job created in this example is a parallel Job with a fixed completion count.

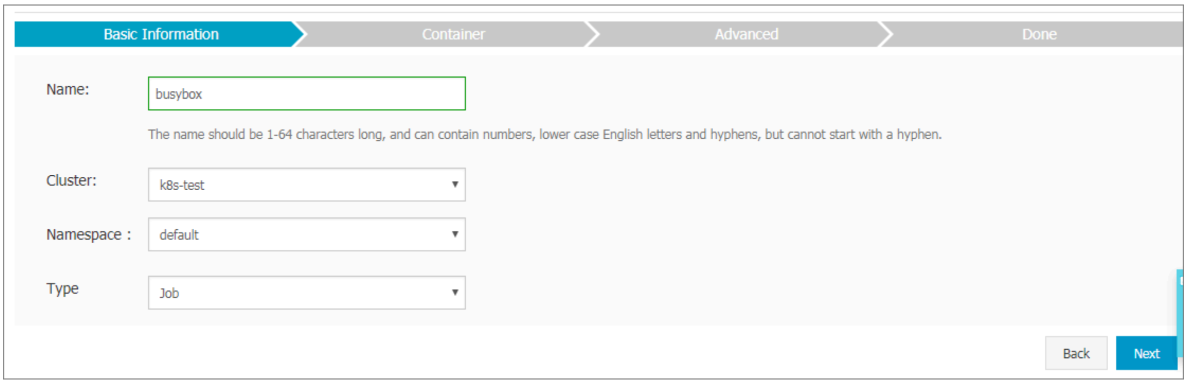
Job pattern	Usage example	Action	Completion	Parallelism
One-off Job	Database migration	A Job creates a pod and the Job is completed when the pod terminates successfully.	1	1
Job with a fixed completion count	Pod that processes the work queue	A Job creates pods one by one. When the pods terminate successfully and the number of the terminated pods reaches the completions value, the Job is completed.	2+	1
Parallel Job with a fixed completion count	Multiple pods process work queues at the same time	A Job creates pods one by one. When the number of pods reaches the completions value, the Job is completed.	2+	2+

Job pattern	Usage example	Action	Completion	Parallelism
Parallel Job	Multiple pods process work queues at the same time	A Job creates one or multiple pods. When at least one pod terminates successfully, the Job is completed.	1	2+

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Application > Job, and then click Create by Image in the upper-right corner.
3. Set the basic parameters and then click Next.
    - Name: Enter a name for the application.
    - Cluster: Select a cluster to which the application is deployed.
    - Namespace: Select a namespace in which the application deployment is located. You can also choose to use the default namespace.
    - Type: Select the Job type.

    Note:
    In this example, select the Job type.



4. Configure containers.

    Note:

You can configure multiple containers for the pods of the application.

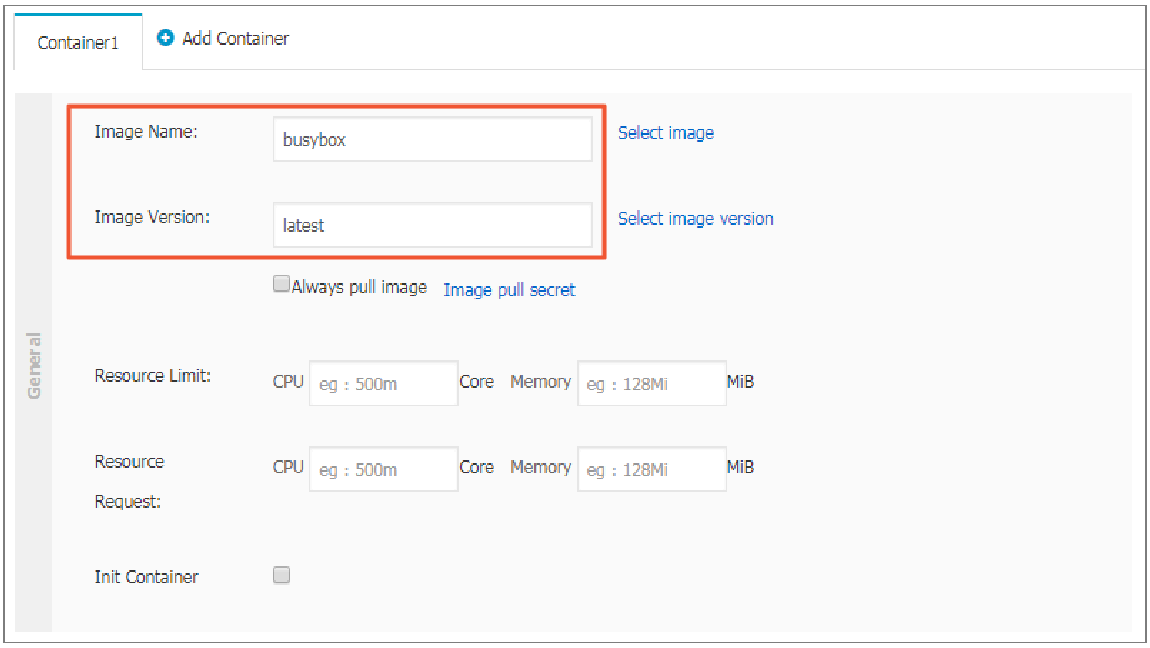
a) Set the container parameters.

- Image Name: Click Select image to select an image in the displayed dialog box and then click OK. In this example, select the busybox image.

  You can also enter a private registry in the format of `domainname / namespace / imagename : tag` to specify an image.
- Image Version: Click Select image version to select a version. If you do not specify any image version, the system uses the latest version by default.
- Always pull image: Container Service caches the image to improve deployment efficiency. During deployment, if the tag of the newly specified image is the same as that of the cached image, Container Service reuses the cached image rather than pulls the same image again. Therefore, if you do not modify the image tag during scenarios where you are changing your code and image, the earlier image in the local cache is used in the application deployment. If you select this check box, Container Service ignores the cached image and re-pulls the image when deploying the application to make sure the latest image and code are always used.
- Image pull secret: If you use a private image, we recommend that you use a secret to guarantee the security of your image. For more information, see Use an image Secret.
- Resource Limit: Specify the upper limit for the resources (CPU and memory) that can be used by this application to avoid occupying excessive resources. CPU is measured in millicores, that is, one thousandth of one core. Memory is measured in bytes, which can be Gi, Mi, or Ki.
- Resource Request: Specify how many resources (CPU and memory) are reserved for the application (that is, these resources become exclusive to the container). If you do not set this parameter, other services or processes will

compete for resources, which means the application may become unavailable due to resource shortage.

- Init Container: Select this check box to create an Init Container that contains useful tools. For more information, see https://kubernetes.io/docs/concepts/workloads/pods/init-containers/.

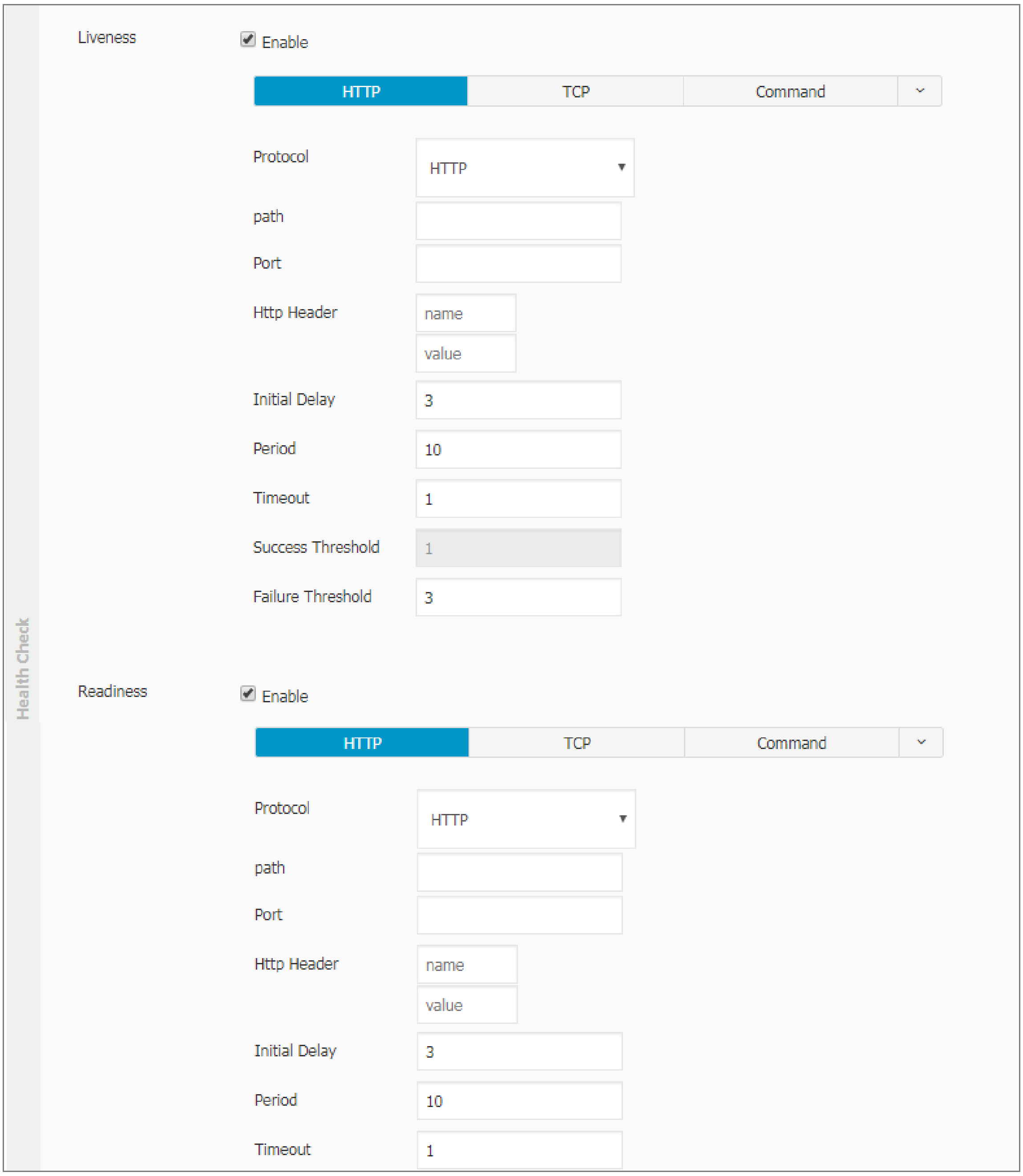


b) Optional: Set Environment .

You can use key-value pairs to set environment variables for the pods. Environment variables are used to add environment labels or pass configurations for the pods. For more information, see Pod variable.

c) Optional: Set Health Check.

You can set liveness probes and readiness probes. Liveness probes are used to detect when to restart the container. Readiness probes determine if the container is ready to receive traffic. For more information about health check, see https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes.



Request method	Description
HTTP request	With this health check method, you can send an HTTP GET request to the container. The following parameters are supported: · Protocol: HTTP/HTTPS. · Path: path to access the HTTP server. · Port: number or name of the access port exposed by the container. The port number must be in the range of 1 to 65535. · HTTP Header: custom headers in the HTTP request. HTTP allows repeated headers. You can use a key-value pair to set an HTTP Header. · Initial Delay (in seconds): namely, the initialDelaySeconds, indicating the number of seconds for which the first probe must wait after the container is started. The default value is 3. · Period (in seconds): namely, the periodseconds, indicating the interval at which probes are performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): namely, the timeoutSeconds, indicating the number of time that the probe has timed out. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes needed for determining a probe success after a failed probe. The default value is 1 and the minimum value is 1. It must be set to 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes needed for determining a probe failure after a successful probe. The default value is 3. The minimum value is 1.

Request method	Description
TCP connection	If you use this health check method, a TCP socket is sent to the container. The kubelet then attempts to open the socket of the container on a specified port. If a connection can be established, the container is considered healthy. If not, it is considered unhealthy. The following parameters are supported: · Port: number or name of the access port exposed by the container. The port number must be in the range of 1 to 65535. · Initial Delay (in seconds): namely, the initialDelaySeconds, indicating the seconds for the first liveness or readiness probe must wait for after the container is started. The default is is 15. · Period (in seconds): namely, the periodseconds, indicating the interval at which probes are performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): namely, the timeoutSeconds, indicating the number of time that the probe has timed out. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes needed for determining a probe success after a failed probe . The default value is 1 and the minimum value is 1. It must be set to 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes needed for determining a probe failure after a successful probe. The default value is 3. The minimum value is 1.

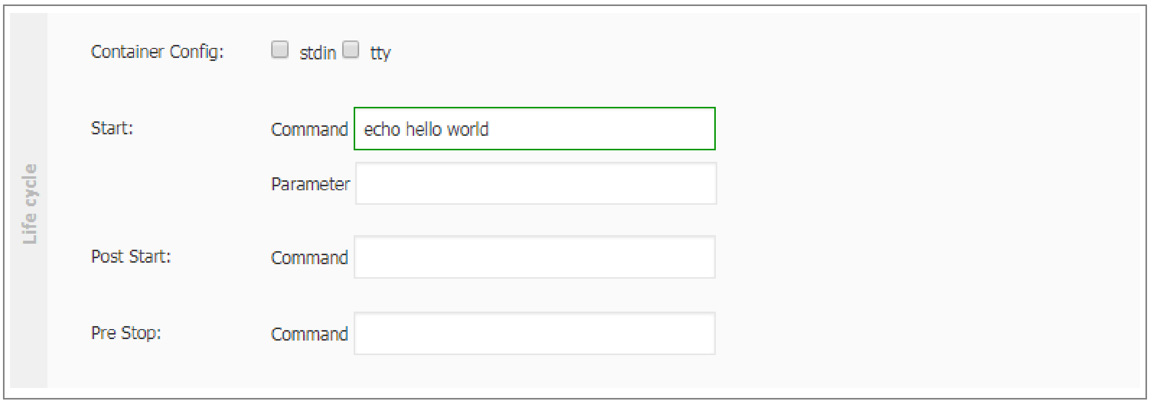
Request method	Description
Command line	With this heath check method, you can detect the container health by executing a probe detection command in the container. The following parameters are supported: · Command: a probe command used to detect the health of the container . · Initial Delay (in seconds): namely, the initialDelaySeconds, indicating the number of seconds for which the first liveness or readiness probe must wait after the container is started. The default value is 5. · Period (in seconds): namely, the periodseconds, indicating the interval at which probes are performed. The default value is 10. The minimum value 1. · Timeout (in seconds): namely, the timeoutSeconds, indicating the number of time that the probe has timed out. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes needed for determining a probe success after a failed probe . The default value is 1 and the minimum value is 1. It must be set to 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes needed for determining a probe failure after a successful probe. The default value is 3. The minimum value is 1.

d) Optional: Set the life cycle.

You can set the following parameters for the container life cycle: container config, start, post start, and pre-stop. For more information, see https://

kubernetes.io/docs/tasks/configure-pod-container/attach-handler-lifecycle-event/.

- Container Config: You can select the stdin check box to enable standard input for the container, or select the tty check box to assign a virtual terminal to the container to send signals to the container. You can also select the two options at the same time. That is, you can bind the terminal (tty) to the container standard input (stdin). For example, an interactive program can obtain standard input from you and then display the obtained standard input in the terminal.
- Start: Set a pre-start command and parameter for the container.
- Post Start: Set a post-start command for the container.
- Pre Stop: Set a pre-stop command for the container.



e) Optional: Set data volumes.

You can configure local storage and cloud storage.

- Local storage: Supported storage types include HostPath, ConfigMap, Secret, and EmptyDir. By setting a type of local storage, you can mount its mount source to the container path. For more information, see Volumes.
- Cloud storage: Supported types of cloud storage include cloud disks, Network Attached Storage (NAS), and Object Storage Service (OSS).

f) Optional: Set Log Service. You can set collection parameters and customize tags.

Note:

Make sure that you have deployed a Kubernetes cluster and installed the log plugin on the cluster.

Set the following log collection parameters:

- Log Store: Set a Logstore. After you specify the Logstore name, the Logstore is generated in Log Service to store collected logs.
- Log path in the container: You can set this parameter to stdout or set a log path.
  - stdout: If you set a log path to stdout, you can collect the standard output logs of the container.
  - text log: If you set a container log path, you can collect the text logs of the path. Wildcards can be used in setting the log file name for a log path.

You can also set custom tags. The custom tags are collected to the container output logs. A custom tag can help you tag container logs, making it easy to collect log statistics, filter logs, and analyze logs by using other methods.

5. After you complete the container configuration, click Next.
6. Configure advanced settings.

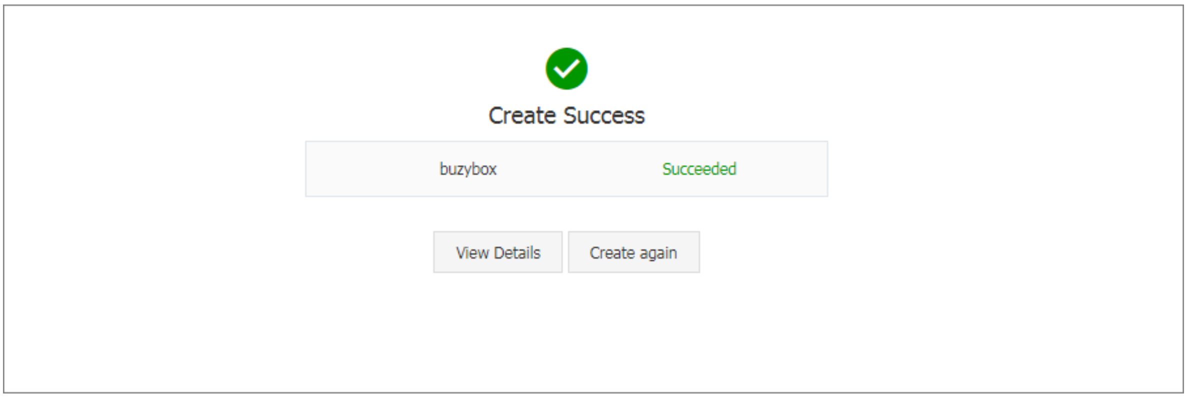
   You can configure Job Settings.

Parameter	Description
Completions	Number of pods that must be run successfully by the configured Job. The default value is 1.
Parallelism	Number of pods that must be run in parallel by the configured Job at any time. The default value is 1.
ActiveDeadlineSeconds	Operating time limit of the configured Job. If the Job is not completed within the time limit, the system tries to terminate the Job.

Parameter	Description
BackoffLimit	Number of retries performed by the configured Job to create pods after a failure. The default is 6. Each time the Job fails, the failed pods associated with the Job are recreated with time delay. The time delay grows exponentially each time. The upper limit of the time delay is six minutes.
Restart	Only Never and OnFailure restart policies are supported.

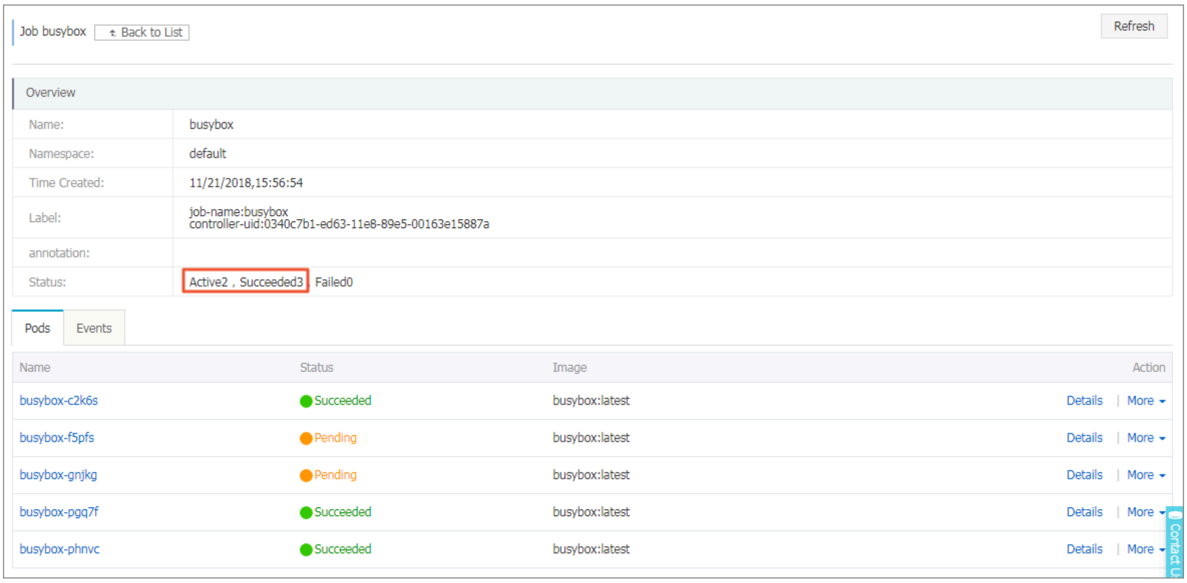
7. Click Create.

8. After you create the Job application, a new page is displayed by default to prompt that you have created the application with the objects included.

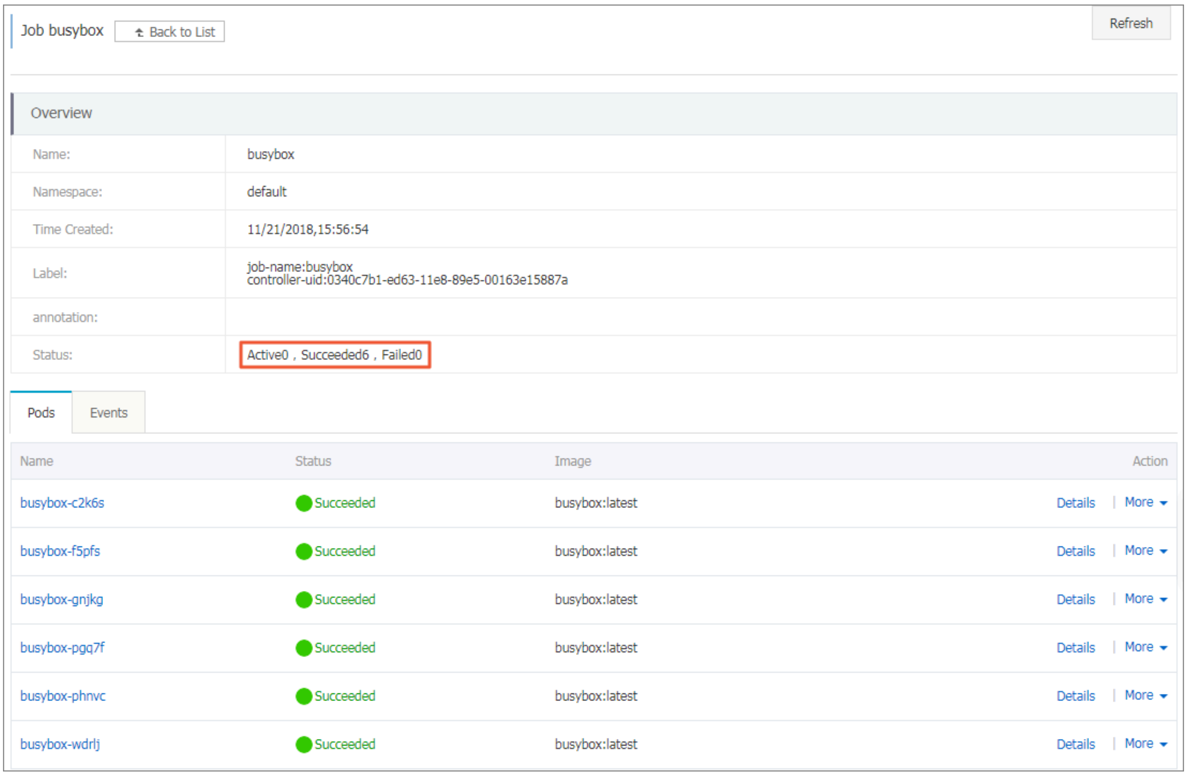


You can click View Details to view the Job details.

During the creation process, you can view the creation status of the pods in the Status column. In this example, two pods are created in parallel according to the Job definition.

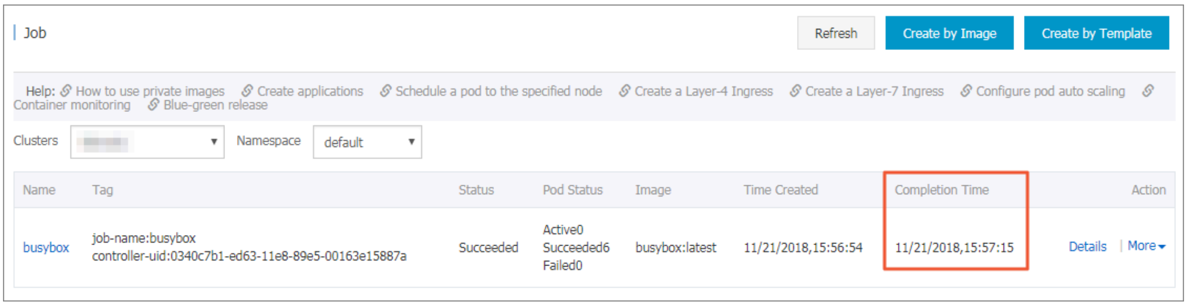


Wait until all pods are created.



9. In the upper-left corner, click Back to List. On the Jog page, the Job completion time is displayed.

> Note:
> If the Job has not created all the pods, the page does not display the Job completion time.



4.4 Create an application in Kubernetes dashboard

You can create an application in the Kubernetes dashboard.

Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Clusters in the left-side navigation pane.

3. Click Dashboard at the right of the cluster to enter the Kubernetes dashboard.

4. In the Kubernetes dashboard, click CREATE in the upper-right corner to create an application.

5. The Resource creation page appears. Configure the application information.

   Create an application in any of the following three ways:

   - CREATE FROM TEXT INPUT: Directly enter the orchestration codes in the YAML or JSON format to create an application. You must know the corresponding orchestration format.

   - CREATE AN APP: Complete the following configurations to create an application.
     - App name: Enter the name of the application you are about to create. In this example, enter `nginx - test`.
     - Container image: Enter the URL of the image to be used. In this example, use Docker Nginx.
     - Number of pods: Configure the number of pods for this application.
     - Service: Select External or Internal. External indicates to create a service that can be accessed from outside the cluster. Internal indicates to create a service that can be accessed from within the cluster.
     - Advanced options: To configure the information such as labels and environment variables, click SHOW ADVANCED OPTIONS. This configuration distributes the traffic load evenly to three pods.

   - CREATE FROM FILE: Upload an existing YAML or JSON configuration file to create an application.
6. Click UPLOAD or DEPLOY to deploy the containers and services.

   You can also click SHOW ADVANCED OPTIONS to configure more parameters.

What's next

After clicking UPLOAD or DEPLOY, you can view the services and containers of the application.

Click Pods in the left-side navigation pane. You can check the status of each Kubernetes object according to the icon on the left. indicates the object is still being deployed. indicates the object has completed the deployment.

4.5 Create a Linux application by using an orchestration template

In a Container Service Kubernetes orchestration template, you must define resource objects required for running an application, and combine the resource objects into a complete application by using label selector.

Prerequisites

A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.

Context

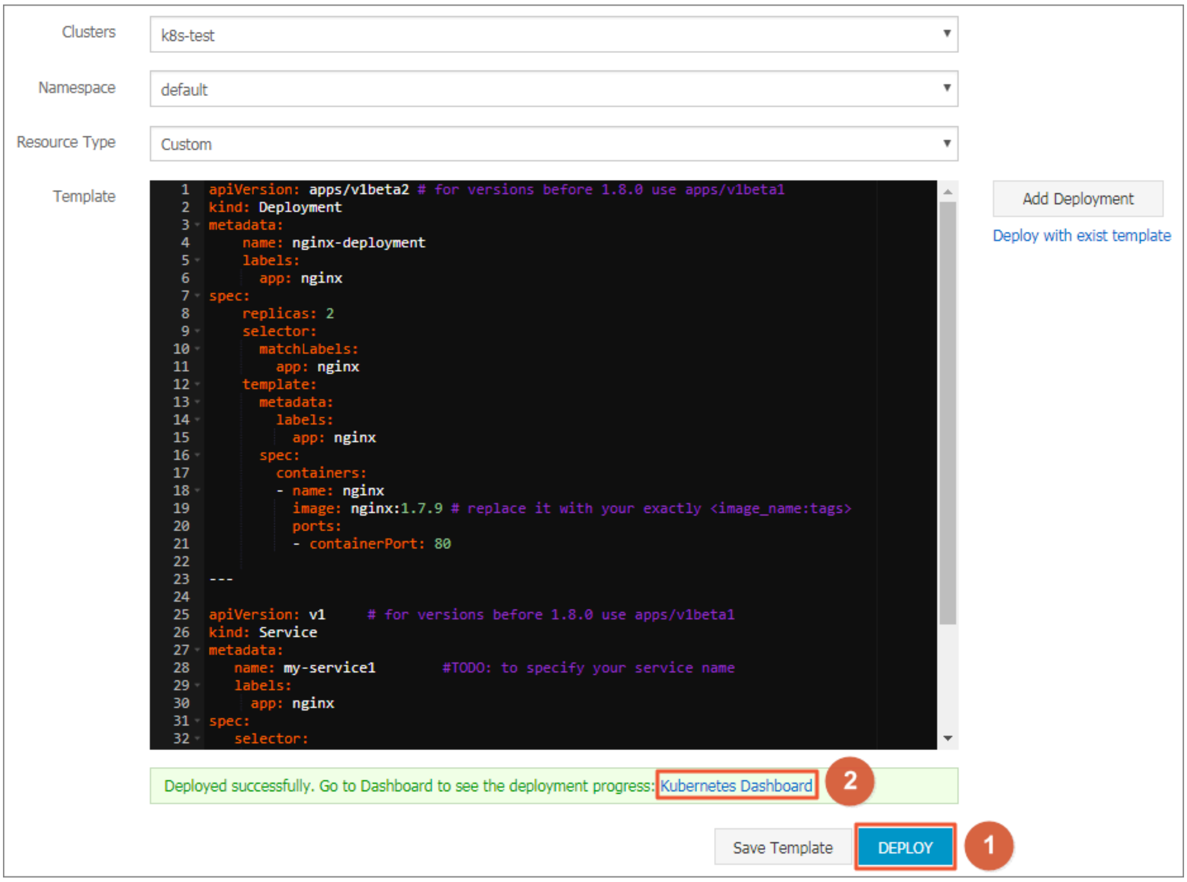
Create an Nginx application in this example. Firstly, create a backend pod resource object by creating the deployment. Then, deploy the service to bind it to the backend pod, forming a complete Nginx application.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments.
3. In the upper-right corner, click Create by Template.

4. Configure the template and then click DEPLOY.

- Clusters: Select the cluster in which in which the resource object is to be deployed.
- Namespace: Select a namespace to which resource object belongs. The default namespace is default. Except for the underlying computing resources such as nodes and persistent storage volumes, most of the resource objects must act on a namespace.
- Resource Type: Alibaba Cloud Container Service provides Kubernetes YAML sample templates of many resource types for you to deploy resource objects quickly. You can write your own template based on the format requirements of Kubernetes YAML orchestration to describe the resource type you want to define.
- Add Deployment: You can quickly define a YAML template with this feature.
- Deploy with exist template: You can import an existing template into the template configuration page.



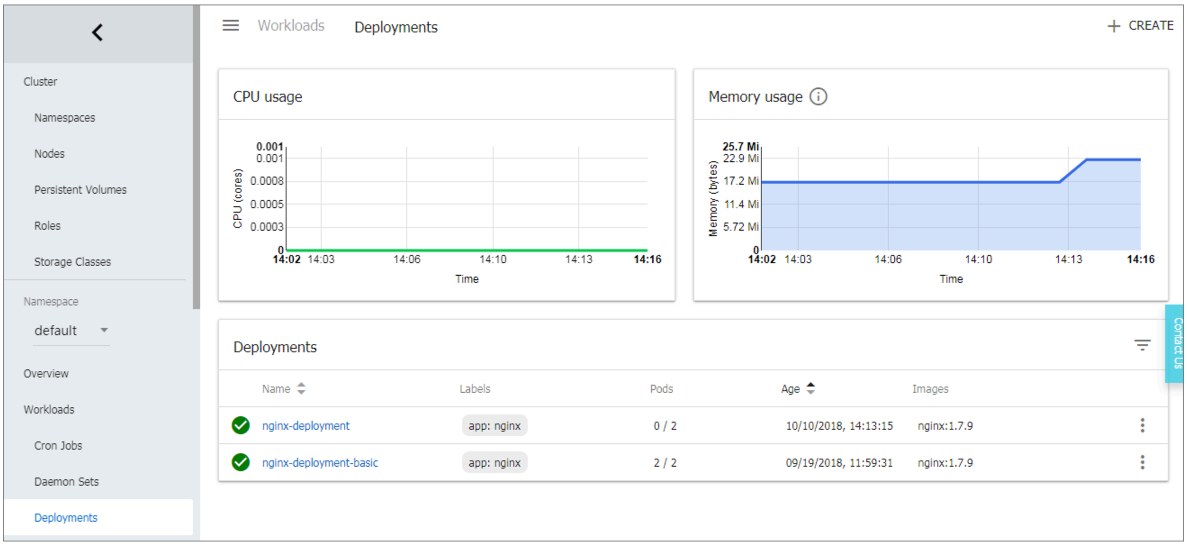
The following is a sample orchestration for an Nginx application. The orchestration is based on an orchestration template built in Container Service. By using this orchestration template, you can create a deployment that belongs to an Nginx application quickly.

Note:
Container Service supports Kubernetes YAML orchestration in which you can use the `---` symbol to separate resource objects so as to create multiple resource objects through a single template.

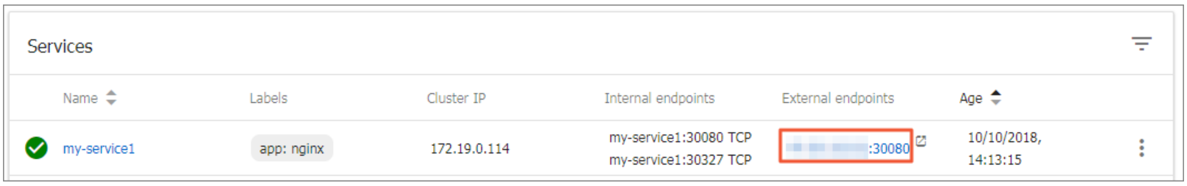
```
apiVersion : apps / v1beta2  #  for  versions  before  1 . 8 .
0  use  apps / v1beta1
kind : Deployment
metadata :
    name : nginx - deployment
    labels :
      app : nginx
spec :
    replicas : 2
    selector :
      matchLabel  s :
        app : nginx
    template :
      metadata :
        labels :
          app : nginx
      spec :
        containers :
       - name : nginx
          image : nginx : 1 . 7 . 9  #  replace  it  with  your
  exactly  < image_name : tags >
          ports :
         - containerP  ort : 80

---

apiVersion : v1     #  for  versions  before  1 . 8 . 0  use
  apps / v1beta1
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 :  LoadBalanc  er          ## In   this   example ,  change
   the   type   from   NodePort   to   LoadBalanc  er .
```

5. After you click DEPLOY, a message indicating the deployment status is displayed. After the deployment succeeds, click Kubernetes Dashboard in the message to go to the dashboard and check the deployment progress.



6. In the Kubernetes dashboard, you can see that the service named my-service1 is successfully deployed and its external endpoint is exposed. Click the access address under External endpoints.



7. You can access the Nginx service welcome page in the browser.

What's next

You can also go back to the home page of Container Services. Then, in the left-side navigation pane under Container Service-Kubernetes, choose Discovery and Load Balancing > Services to view the Nginx service.

4.6 Create a Windows application by using an orchestration template

This topic describes how to create a Windows application by using an orchestration template. Such a template is used to customize the resources required by a Windows application to operate.

Prerequisites

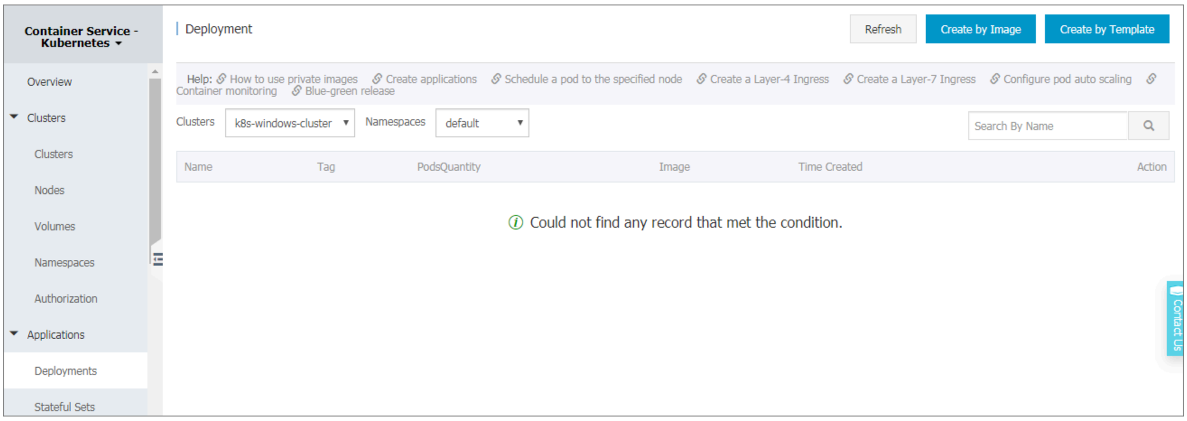
A Kubernetes cluster that supports Windows is created. For more information, see Create a Windows application by using an orchestration template.

Context

In this topic, an application named `aspnet` is created by using an orchestration template. This application contains a deployment and a service. On the backend, the deployment creates pods according to settings. Then, the service is associated with the pods.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Deployment.
3. In the upper-right corner, click Create by Template.



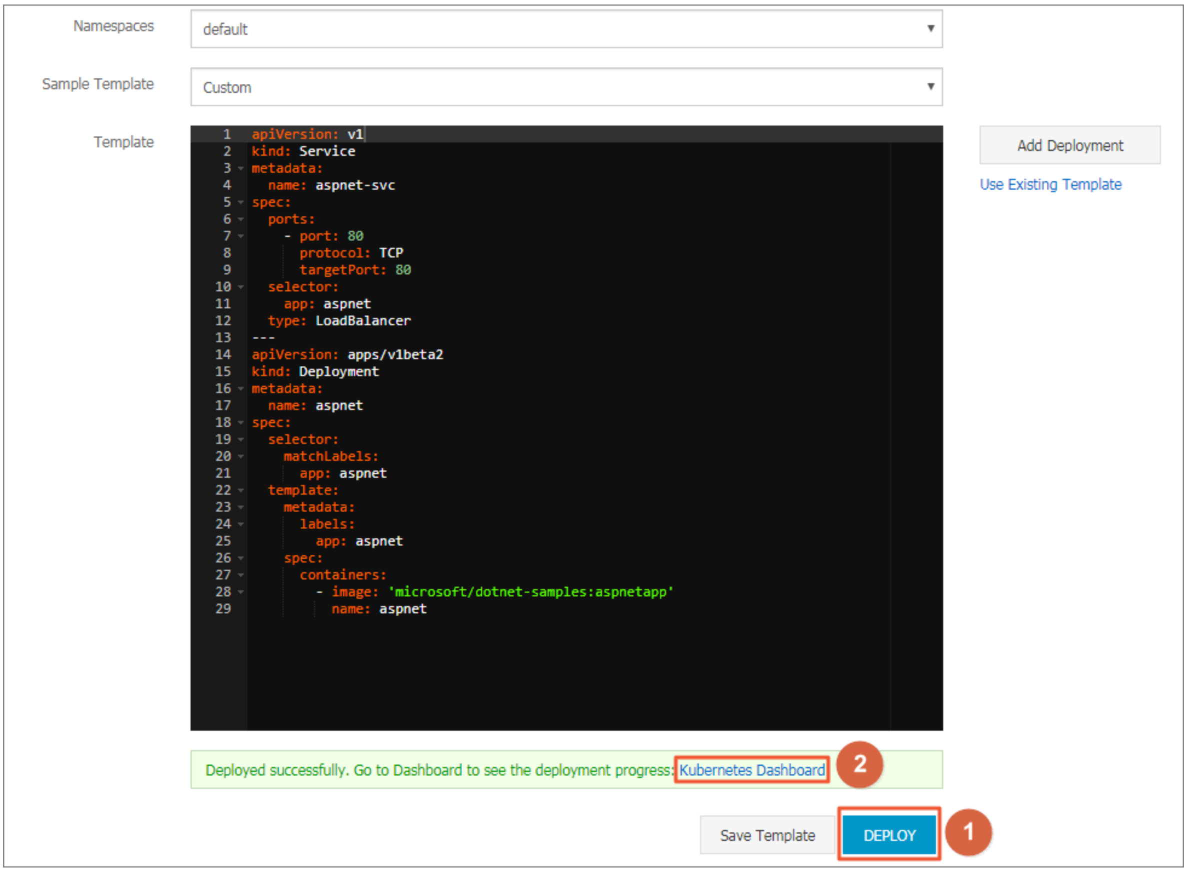
4. To set the orchestration template, set the following:

    - Clusters: Select the target cluster. The resources required by the application are deployed in a cluster.
    - Namespace: Select the target namespace. The default namespace is preset. Except for nodes, persistent volumes, and other underlying resource types, most resources required by the application are deployed in a namespace.
    - Sample Template: Select the target sample template. Alibaba Cloud Container Service for Kubernetes provides many built-in YAML orchestration templates for

different types of resources. You can customize an orchestration template to set a type of resource according to the YAML orchestration requirements.

- Add Deployment: Edit a YAML template quickly by using this function.
- Using Existing Template: Import an existing template to the template setting area.

Then, click DEPLOY.



The following is an orchestration template for the Windows application named `aspnet`. With such an orchestration template, you can quickly create a deployment and a service for the application.

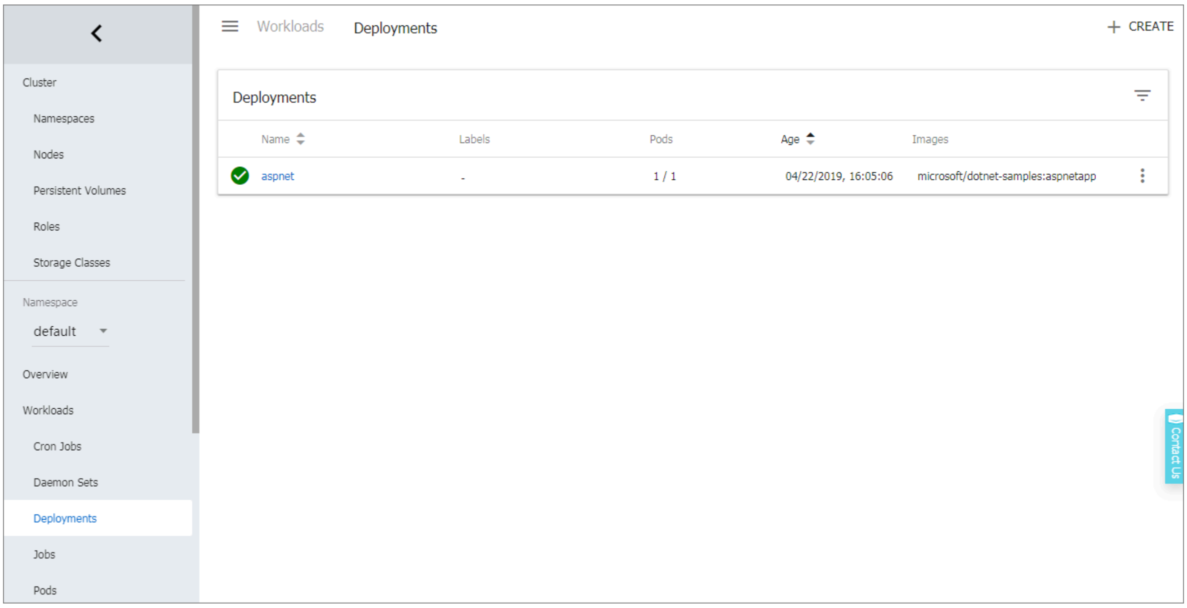
Note:

If you want to create multiple resources in a template, you can use `---` to separate different resources.

```
apiVersion :  v1
kind :  Service
metadata :
  name :  aspnet - svc
spec :
  ports :
   -  port :  80
```

```
        protocol :  TCP
        targetPort :  80
    selector :
      app :  aspnet
    type :  LoadBalanc  er
---
  apiVersion :  apps / v1beta2
  kind :  Deployment
  metadata :
    name :  aspnet
  spec :
    selector :
      matchLabel  s :
        app :  aspnet
    template :
      metadata :
        labels :
          app :  aspnet
      spec :
        containers :
         - image : ' microsoft / dotnet - samples : aspnetapp '
           name :  aspnet
```

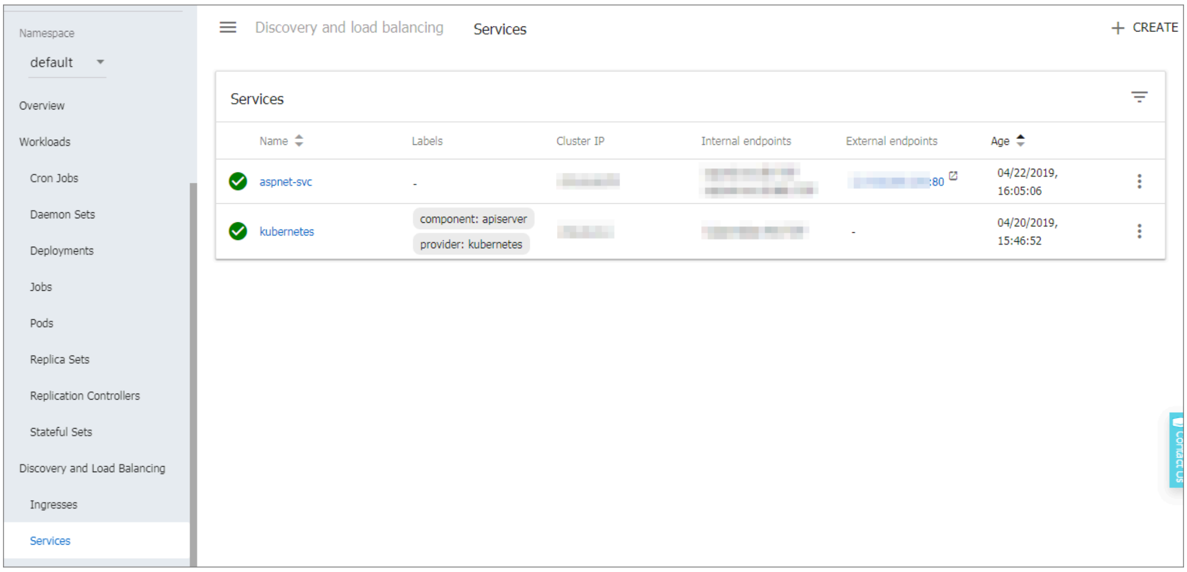
5. A message is displayed on the bottom of the Template area to show the result. If a success message is displayed (shown in the preceding figure), click Kubernetes Dashboard at the end of the message to view the progress.



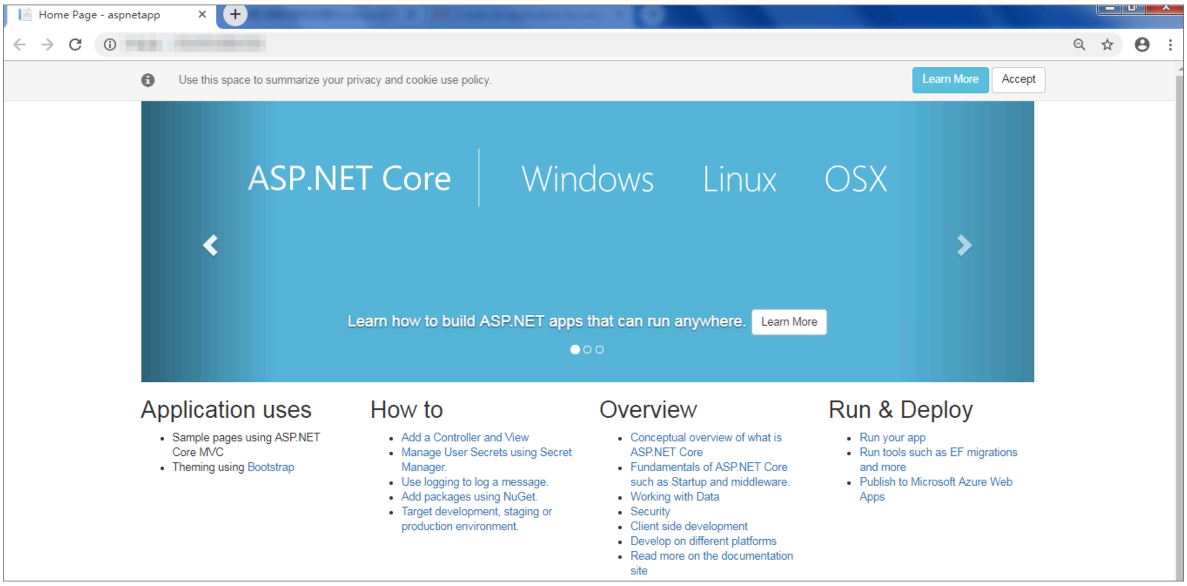
6. In the left-side navigation pane, choose Discovery and Load Balancing > Service. Then, in the External endpoints column, click the IP address of the created service named `aspnet - svc` to visit the home page of the `aspnet` application in your browser.

Note:

In the Kubernetes dashboard, you can view that a service named `aspnet - svc` and its external endpoint are created.



The following figure shows the home page of the `aspnet` application.



What's next

You can also return to the home page of Container Service-Kubernetes, and then choose Discovery and Load Balancing > Service in the left-side navigation pane to view the service of the `aspnet` application.

4.7 Manage applications by using commands

You can create applications or view containers in applications by using commands.

Prerequisites

Before using commands to manage applications, #unique_19.

Create an application by using commands

Run the following statements to run a simple container (a Nginx Web server in this example).

```
 root @ master   #  kubectl   run  - it   nginx  -- image = registry .
aliyuncs . com / spacexnice / netdia : latest
```

This command creates a service portal for this container. Specify `-- type = LoadBalanc er` and an Alibaba Cloud Server Load Balancer route will be created to the Nginx container.

```
root @ master  #  kubectl   expose   deployment   nginx  -- port = 80
 -- target - port = 80  -- type = LoadBalanc  er
```

View containers by using commands

Run the following command to list all the running containers in the default namespaces.

```
root @ master  #  kubectl   get   pods
NAME   READY   STATUS   RESTARTS   AGE
nginx - 2721357637 - dvwq3   1 / 1   Running   1   9h
```

4.8 Simplify Kubernetes application deployment by using Helm

In Kubernetes, app management is the most challenging and in demand field. The Helm project provides a uniform software packaging method which supports version control and greatly simplifies Kubernetes app distribution and deployment complexity.

Alibaba Cloud Container Service integrates the app catalog management function with the Helm tool, extends the functions, and supports official repository, allowing you to deploy the application quickly. You can deploy the application in the Container Service console or by using command lines.

This document introduces the basic concepts and usage of Helm and demonstrat es how to use Helm to deploy the sample applications WordPress and Spark on an Alibaba Cloud Kubernetes cluster.

Basic concepts of Helm

Helm is an open-source tool initiated by Deis and helps to simplify the deployment and management of Kubernetes applications.

You can understand Helm as a Kubernetes package management tool that facilitate s discovery, sharing and use of apps built for Kubernetes. It involves several basic concepts.

- Chart: A Helm package containing the images, dependencies, and resource definitions required for running an application. It may also contain service definitions in a Kubernetes cluster, similar to the formula of Homebrew, the dpkg of APT, or the rpm file of Yum.
- Release: A chart running on a Kubernetes cluster. A chart can be installed multiple times on the same cluster. A new release will be created every time a chart is installed. For example, to run two databases on the server, you can install the MySQL chart twice. Each installation will generate its own release with its own release name.
- Repository: The repository for publishing and storing charts.

Helm components

Helm adopts a client/server architecture composed of the following components:

- Helm CLI is the Helm client and can be run locally or on the master nodes of the Kubernetes cluster.
- Tiller is the server component and runs on the Kubernetes cluster. It manages the lifecycles of Kubernetes applications.
- Repository is the chart repository. The Helm client accesses the chart index files and packages in the repository by means of the HTTP protocol.

Use Helm to deploy applications

Prerequisites

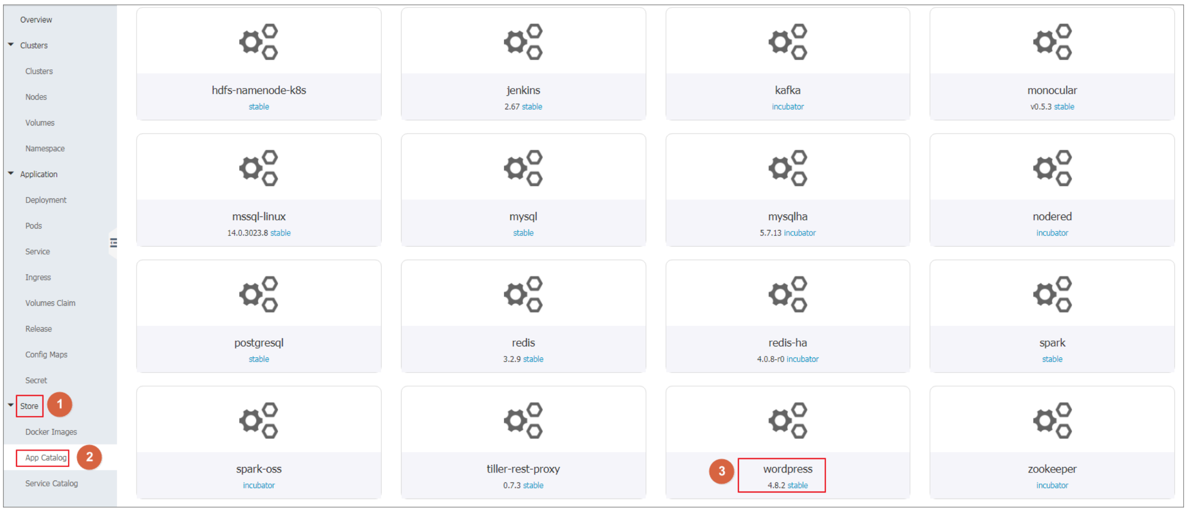
- Before using Helm to deploy an application, create a Kubernetes cluster in Alibaba Cloud Container Service. For more information, see Create a Kubernetes cluster.

  Tiller is automatically deployed to the cluster when the Kubernetes cluster is created. Helm CLI is automatically installed on all the master nodes and the configuration points to the Alibaba Cloud chart repository.
- Check the Kubernetes version of your cluster.

  Only clusters whose Kubernetes version is 1.8.4 or later are supported. For clusters whose Kubernetes version is 1.8.1, upgrade the cluster on the Cluster List page.

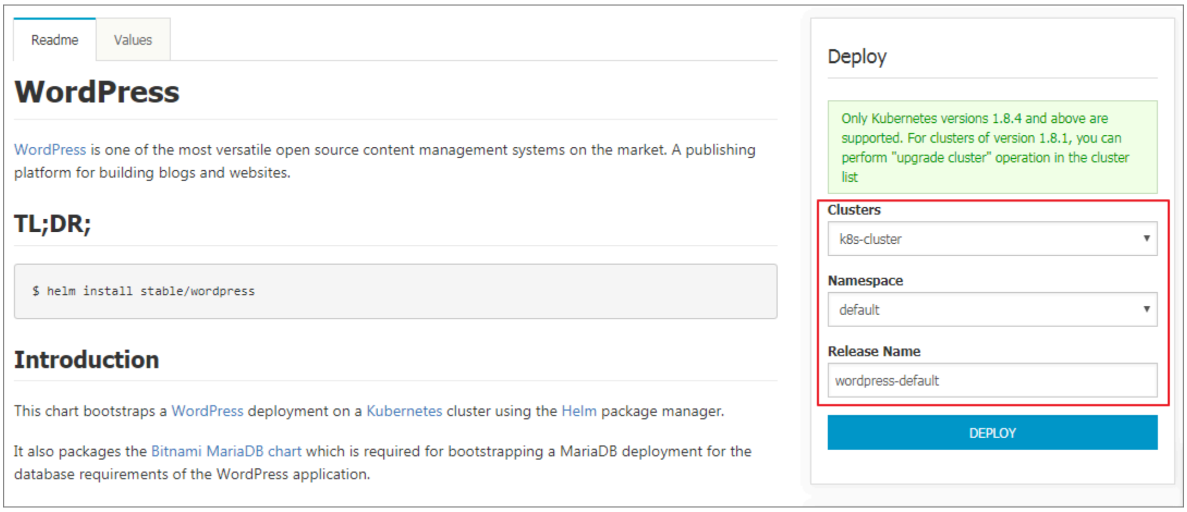
Deploy applications in Container Service console

1. Log on to the Container Service console.
2. Under Kubernetes, click Store > App Catalog in the left-side navigation pane.
3. On the App Catalog page, click a chart (WordPress in this example) to enter the chart details page.



4. Enter the basic information for the deployment on the right.

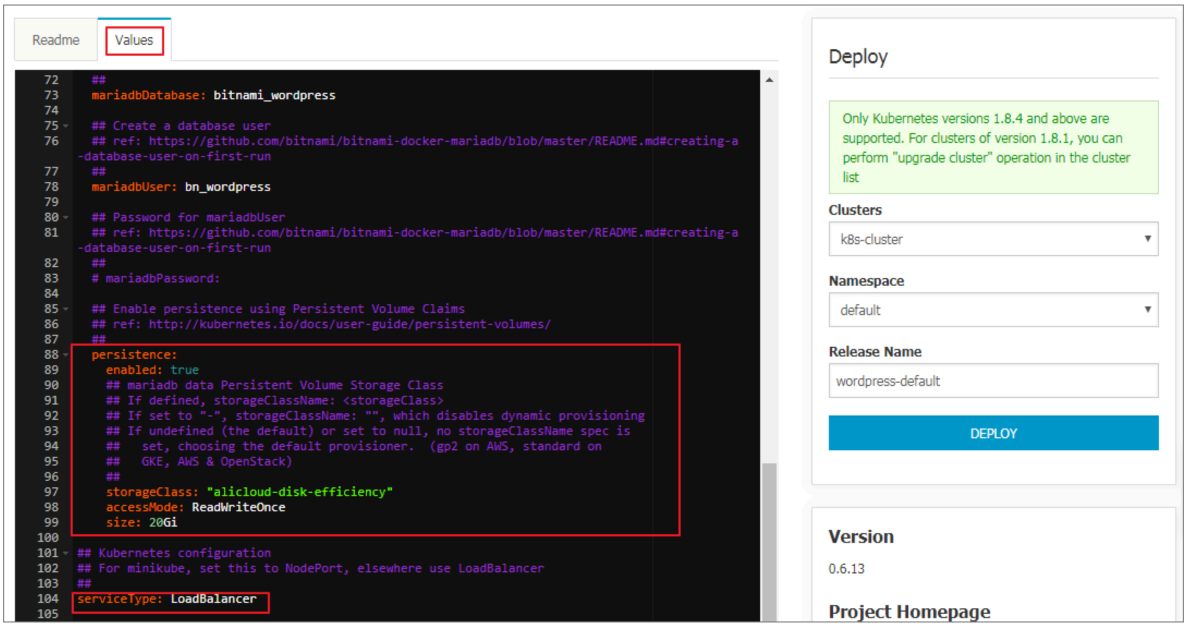
   - Clusters: Select the cluster in which the application is to be deployed.
   - Namespace: Select the namespace. default is selected by default.
   - Release Name: Enter the release name for the application. Enter test in this example.



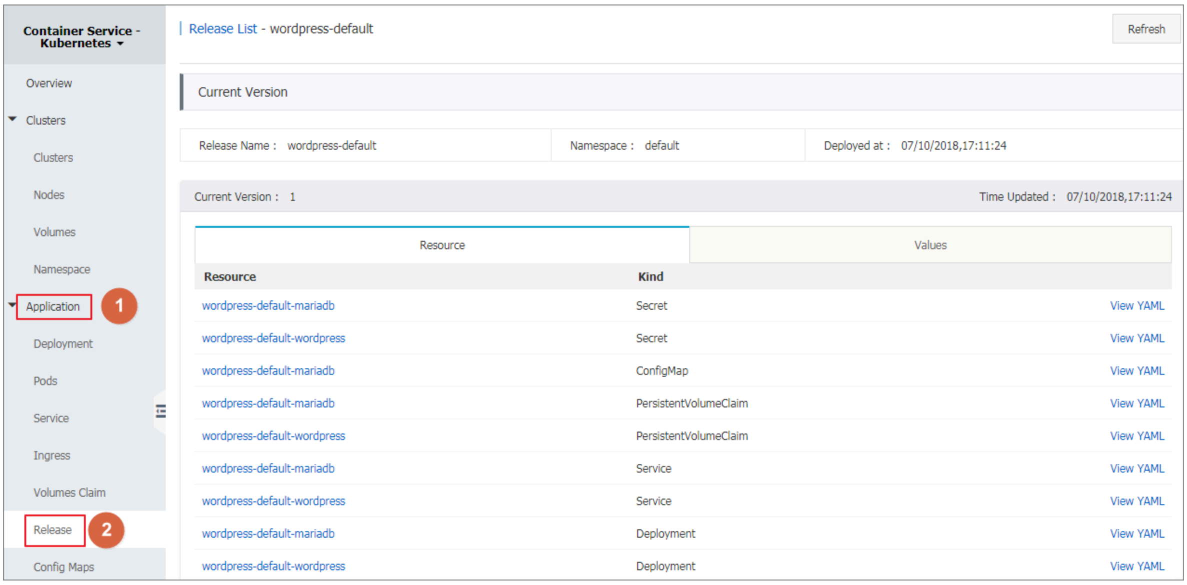
5. Click the Values tab to modify the configurations.

   In this example, bind dynamic data volumes of the cloud disk to a persistent storage volume claim (PVC). For more information, see .

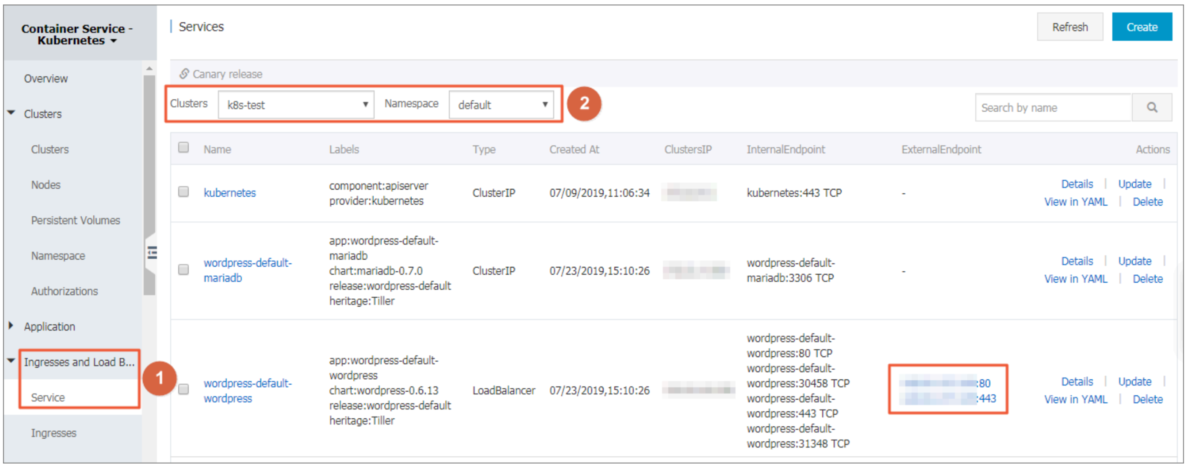
> **Note:**
> You need to create a persistent storage volume (PV) of cloud disk in advance. The capacity of the PV cannot be less than the value defined by the PVC.



6. Click DEPLOY after completing the configurations. After the successful deployment, you are redirected to the release page of this application.



7. Click Ingresses and Load Balancing > Service in the left-hand navigation pane. Select the target cluster and namespace and find the corresponding service. You can obtain the HTTP/HTTPS external endpoint address.



8. Click the preceding access address to enter the WordPress blog publishing page.

Deploy applications by using command lines

You can use SSH to log on to the master node of the Kubernetes cluster when deploying applications by using command lines (Helm CLI is automatically installed and has configured the repository). For more information, see Access Kubernetes clusters by using SSH. You can also install and configure the kubectl and Helm CLI locally.

In this example, install and configure the kubectl and Helm CLI locally and deploy the applications WordPress and Spark.

Install and configure kubectl and Helm CLI

1. Install and configure kubectl on a local computer.

   For more information, see Connect to a Kubernetes cluster by using kubectl.

   To view information of the target Kubernetes cluster, enter the command `kubectl cluster - info`.
2. Install Helm on a local computer.

   For the installation method, see Install Helm.
3. Configure the Helm repository. Here the charts repository provided by Alibaba Cloud Container Service is used.

```
helm  init  -- client - only  -- stable - repo - url  https ://
aliacs - app - catalog . oss - cn - hangzhou . aliyuncs . com /
charts /
helm  repo  add  incubator  https :// aliacs - app - catalog .
oss - cn - hangzhou . aliyuncs . com / charts - incubator /
helm  repo  update
```

Basic operations of Helm

- To view the list of charts installed on the cluster, enter the following command:

  `helm list`

  Or you can use the abbreviated version:

  `helm ls`
- To view the repository configurations, enter the following command:

  `helm repo list`
- To view or search for the Helm charts in the repository, enter one of the following commands:

```
helm  search
helm  search  repository  name  # For  example ,  stable  or
incubator .
helm  search  chart  name  # For  example ,  wordpress  or
spark .
```

- To update the chart list to get the latest version, enter the following command:

  `helm repo update`

For more information about how to use Helm, see Helm document.

Deploy WordPress by using Helm

Use Helm to deploy a WordPress blog website.

Enter the following command.

```
helm   install  -- name   wordpress - test   stable / wordpress
```

Note:
The Alibaba Cloud Kubernetes service provides the support for dynamic storage volumes of block storage (cloud disk). You need to create a storage volume of cloud disk in advance.

The result is as follows:

```
NAME :  wordpress - test
LAST   DEPLOYED :  Mon   Nov   20   19 : 01 : 55   2017
NAMESPACE :  default
STATUS :  DEPLOYED
...
```

Use the following command to view the release and service of WordPress.

```
helm   list
kubectl   get   svc
```

Use the following command to view the WordPress related pods and wait until the status changes to Running.

```
kubectl   get   pod
```

Use the following command to obtain the WordPress access address:

```
echo   http ://$( kubectl   get   svc   wordpress - test - wordpress
- o   jsonpath ='{. status . loadBalanc  er . ingress [ 0 ]. ip }')
```

Access the preceding URL in the browser, and you can see the familiar WordPress website.

You can also follow the chart instructions and use the following command to obtain the administrator account and password of the WordPress website:

```
echo   Username :  user
```

```
echo   Password : $( kubectl   get   secret  -- namespace   default
wordpress - test - wordpress  - o   jsonpath ="{. data . wordpress -
password }" |  base64  -- decode )
```

To completely delete the WordPress application, enter the following command:

```
helm   delete  -- purge   wordpress - test
```

Deploy Spark by using Helm

Use Helm to deploy Spark for processing big data.

Enter the following command:

```
helm   install  -- name   myspark   stable / spark
```

The result is as follows:

```
NAME :  myspark
LAST   DEPLOYED :  Mon   Nov   20   19 : 24 : 22   2017
NAMESPACE :  default
STATUS :  DEPLOYED
...
```

Use the following commands to view the release and service of Spark.

```
helm   list
kubectl   get   svc
```

Use the following command to view the Spark related pods and wait until the status changes to Running. Pulling images takes some time because the Spark related images are large.

```
kubectl   get   pod
```

Use the following command to obtain the Spark Web UI access address:

```
echo   http ://$( kubectl   get   svc   myspark - webui  - o
jsonpath ='{. status . loadBalanc  er . ingress [ 0 ]. ip }'): 8080
```

Access the preceding URL in the browser, and you can see the Spark Web UI, on which indicating currently three worker instances exist.

Then, use the following command to use Helm to upgrade the Spark application and change the number of worker instances from three to four. The parameter name is case sensitive.

```
helm   upgrade   myspark  -- set  " Worker . Replicas = 4 "  stable /
spark
```

The result is as follows:

```
Release  " myspark "  has   been   upgraded .  Happy   Helming !
LAST   DEPLOYED :  Mon   Nov   20   19 : 27 : 29   2017
NAMESPACE :  default
STATUS :  DEPLOYED
...
```

Use the following command to view the newly added pods of Spark and wait until the status changes to Running.

```
kubectl   get   pod
```

Refresh the Spark Web UI in the browser. The number of worker instances changes to four.

To completely delete the Spark application, enter the following command:

```
helm   delete  -- purge   myspark
```

Use third-party chart repository

Besides the preset Alibaba Cloud chart repository, you can also use the third-party chart repository (make sure the network is accessible). Add the third-party chart repository in the following command format:

```
helm   repo   add   repository   name   repository   URL
helm   repo   update
```

For more information about the Helm related commands, see Helm document.

References

Helm boosts the growth of communities. More and more software providers, such as Bitnami, have begun to provide high-quality charts. You can search for and discover existing charts at `https :// kubeapps . com /`.

4.9 Use an application trigger to redeploy an application

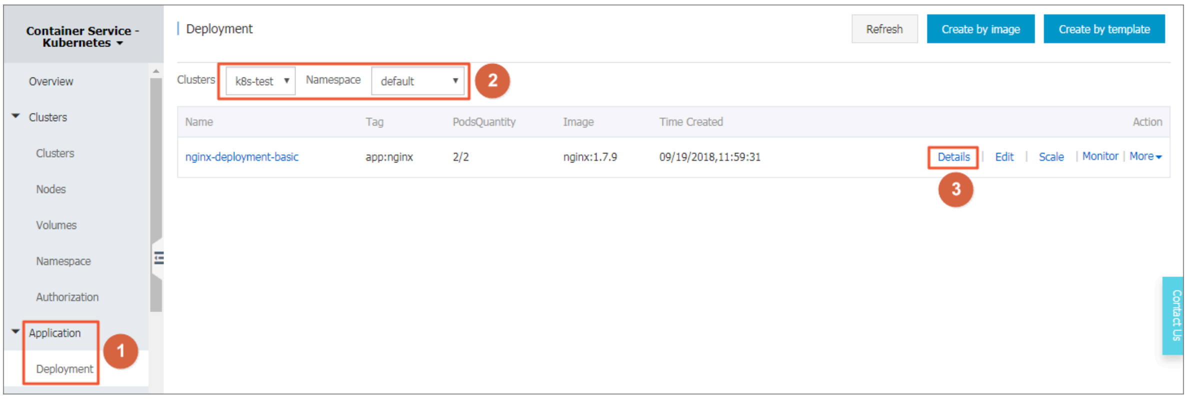
Alibaba Cloud Container Service Kubernetes supports the application trigger function. You can use an application trigger in many ways.

Prerequisites

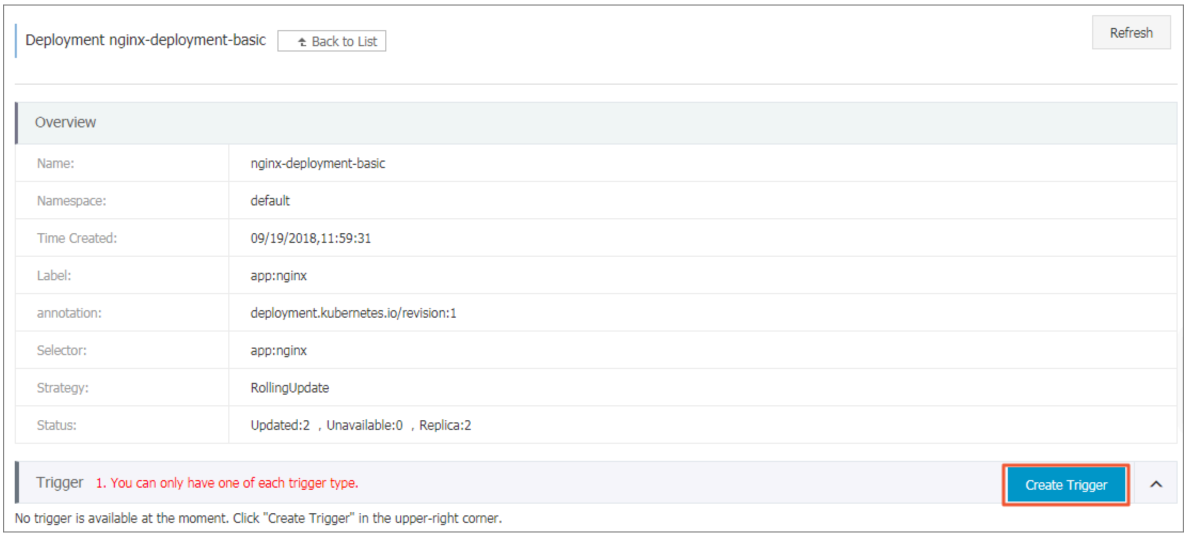
- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have created an application that is used to create an application trigger and test the trigger. In this example, create an nginx application.

Procedure

1. Log on to the Container Service console.
2. Click Application > Deployment and select a cluster and namespace. Click Details at the right of the target nginx application.

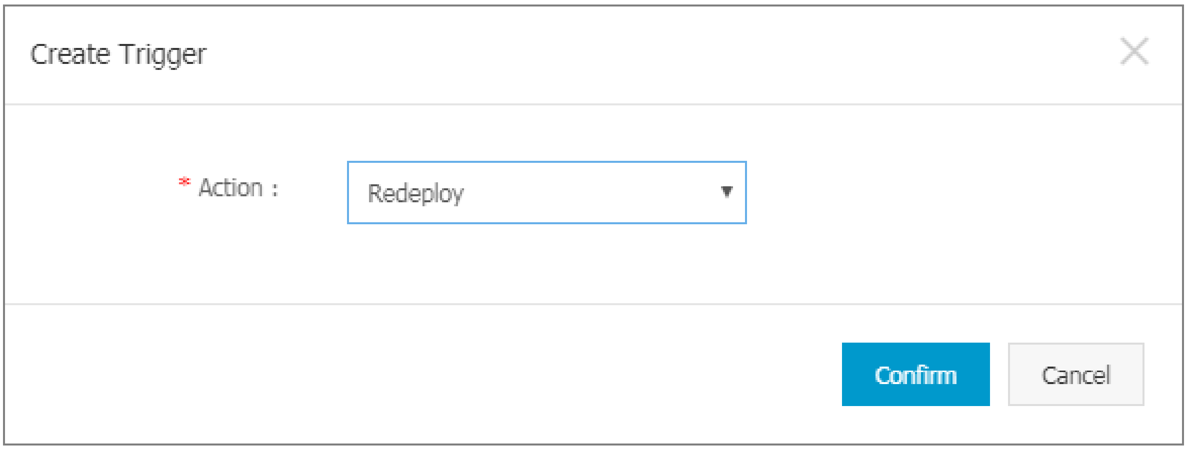


3. On the nginx application details page, click Create Trigger on the right side of the trigger bar.

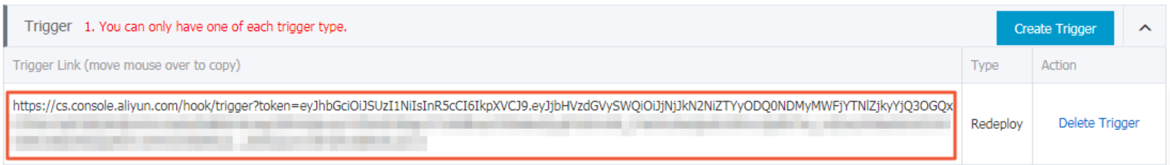


4. In the pop-up dialog box, click Redeploy and click Confirm.

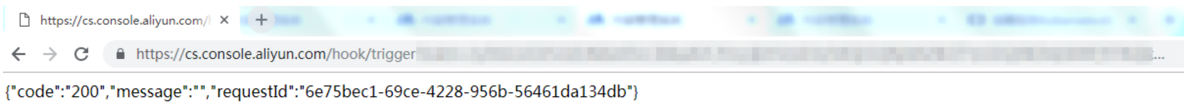
> **Note:**
> Currently, only the redeploy action is supported.



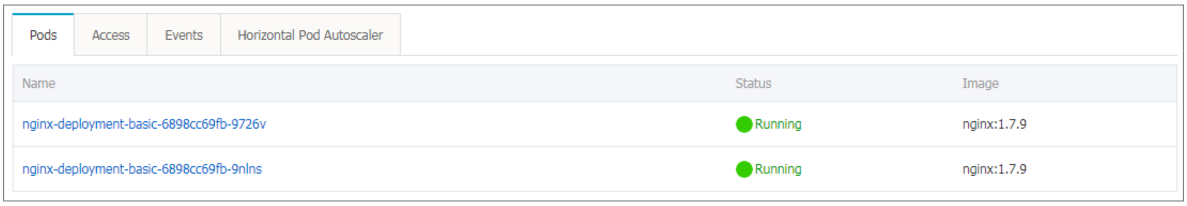
After the trigger is created, a trigger link is displayed in the trigger bar on the nginx application detail page.



5. Copy the trigger link and visit it in the browser. A message is returned on the web page, containing information such as the request ID.



6. Back to the nginx application detail page, you can see that a new pod appears.



After a period of time, the nginx application removes the old pod and keeps only the new pod.

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.

Call the redeploy trigger as follows:

```
curl  https :// cs . console . aliyun . com / hook / trigger ? token =
xxxxxxx
```

4.10 Schedule a pod to a specific node

This topic describes how to schedule a pod to a specific node in the Container Service console.

You can add a node label and then configure the `nodeSelect  or` to schedule a pod to a specified node. For more information about the implementation principle of nodeSelector, see nodeselector.

For business scenario needs, to deploy a service used for management and control to a master node, or deploy services to a machine with an SSD disk, you can use this method to schedule pods to specified nodes.

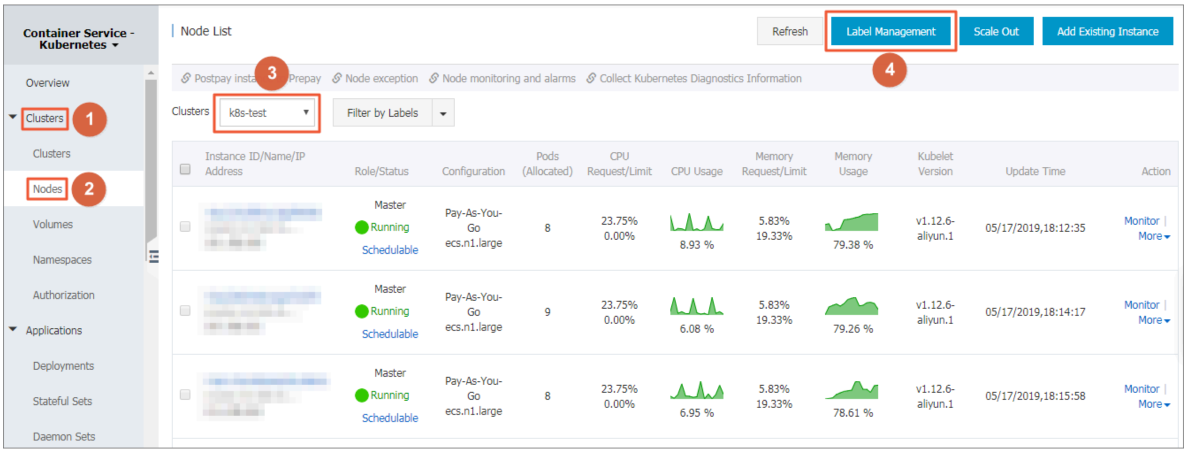
Prerequisites

A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.

Step 1: Add a node label

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Nodes.

3. Select the cluster from the Cluster drop-down list and then click Label Management in the upper-right corner.



4. Select one or more nodes by selecting the corresponding check boxes and then click Add Tag. In this example, select a worker node.

5. Ener the name and value of the label in the displayed dialog box and then click OK.

   The node label `group : worker` is displayed on the Label Management page.

You can also add a node label by running the command `kubectl label nodes < node - name > < label - key >=< label - value >`.

Step 2: Deploy a pod to a specified node

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments.
3. In the upper-right corner, click Create by Template.

4. Configure the template to deploy a pod. After completing the configurations, click DEPLOY.

   - Clusters: Select a cluster.
   - Namespace: Select the namespace to which the resource object belongs. In this example, use default as the namespace.
   - Resource Type: Select Custom in this example.

   The orchestration template in this example is as follows:

```
apiVersion :  v1
 kind :  Pod
 metadata :
   labels :
     name :  hello - pod
   name :  hello - pod
 spec :
   containers :
    -  image :  nginx
       imagePullP  olicy :  IfNotPrese  nt
       name :  hello - pod
       ports :
        -  containerP  ort :  8080
           protocol :  TCP
       resources : {}
       securityCo  ntext :
         capabiliti  es : {}
         privileged :  false
       terminatio  nMessagePa  th : / dev / terminatio  n - log
   dnsPolicy :  ClusterFir  st
   restartPol  icy :  Always
   nodeSelect  or :
     group :  worker   ## The   same   as   the   node   label
configured   in   the   preceding   step .
 status  :{}
```

5. A message indicating the deployment status is displayed after you click DEPLOY. After the successful deployment, click Kubernetes Dashboard in the message to go to the dashboard and check the deployment status.

6. Click the pod name to view the pod details.

   You can view the information such as the pod label and node ID, which indicates the pod is successfully deployed to a node with the label `group : worker`.

4.11 View the pods of a Kubernetes cluster

This topic describes how to view the pods of a Kubernetes cluster on the undefinedPod page or undefinedDashboard page.

Procedure

To view a pod on the Pod page or Dashboard page, follow these steps:

View pods of a Kubernetes cluster on the Pod page

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Pods.
3. Select the target cluster and namespace, and find the target pod. Then, on the right of the target pod, click Details.

   Then, you can view the pod details.

View pods of a Kubernetes cluster on the Dashboard page

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.
3. Find the target cluster. Then, in the Action column, click Dashboard.
4. In the left-side navigation pane, click Pods to view pods of the target cluster.

   Note:
   You can also click Services in the left-side navigation pane, and then click a service name to view the pods in this service.

   Note:
   In the Pods area, the icon on the left of each pod shows the pod status.

   - indicates the pod is still being deployed.

- indicates the pod is deployed.

5. Click the name of a pod to view the details, CPU usage, and memory usage of the pod.
6. In the upper-right corner, click LOGS to view the pod logs.
7. You can also click the icon on the right of the pod, and then select Delete to delete the pod.

4.12 Change container configurations

You can change the container configurations in the Container Service console.

Procedure

1. Log on to the Container Service console.
2. Click Kubernetes > Clusters in the left-side navigation pane.
3. Click Dashboard at the right of the cluster to enter the Kubernetes dashboard.
4. In the Kubernetes dashboard, click Pods in the left-side navigation pane.
5. Click the icon at the right of the pod and then select View/edit YAML.
6. The Edit a Pod dialog box appears. Change the container configurations and then click UPDATE.

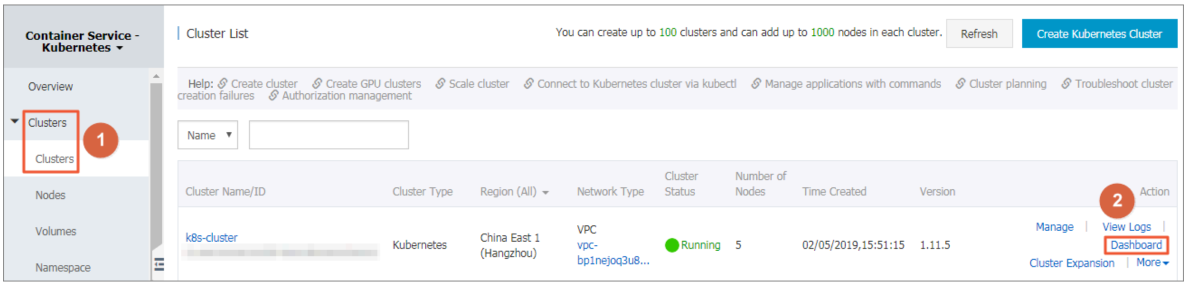
4.13 Scale a service

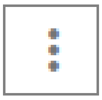
This topic describes how to scale out or scale in an application service as needed after an application is created.

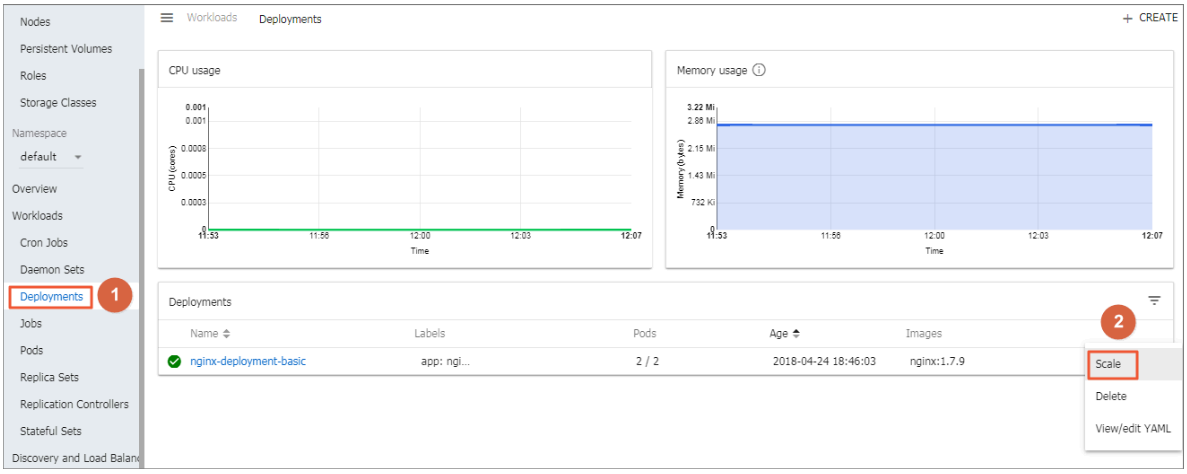
Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Clusters.

3. On the right of the target cluster, click Dashboard.

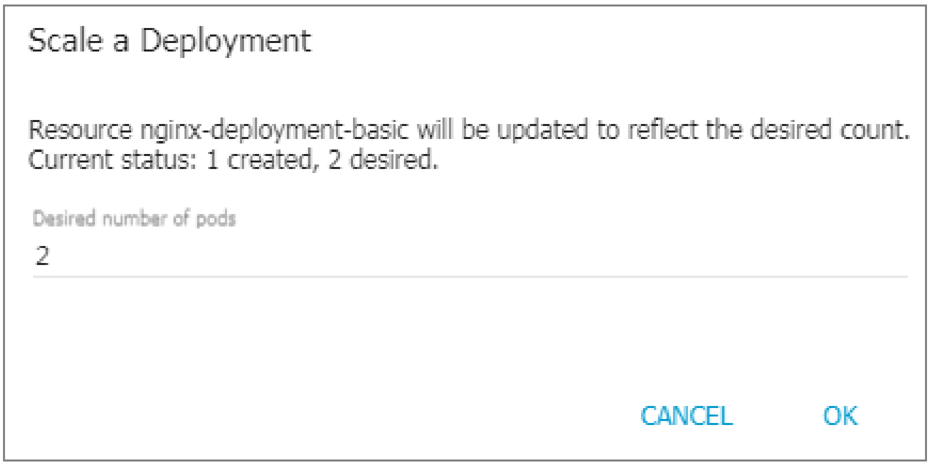


4. In the left-side navigation pane, click Deployments.
5. Click the  icon on the right of the target deployment, and then click Scale.



6. In the displayed dialog box, change the value of Desired number of pods to the number you require. Here, the example number of desired pods is 2. Then, click OK.

   This action adds a new pod. The number of replicas becomes 2.

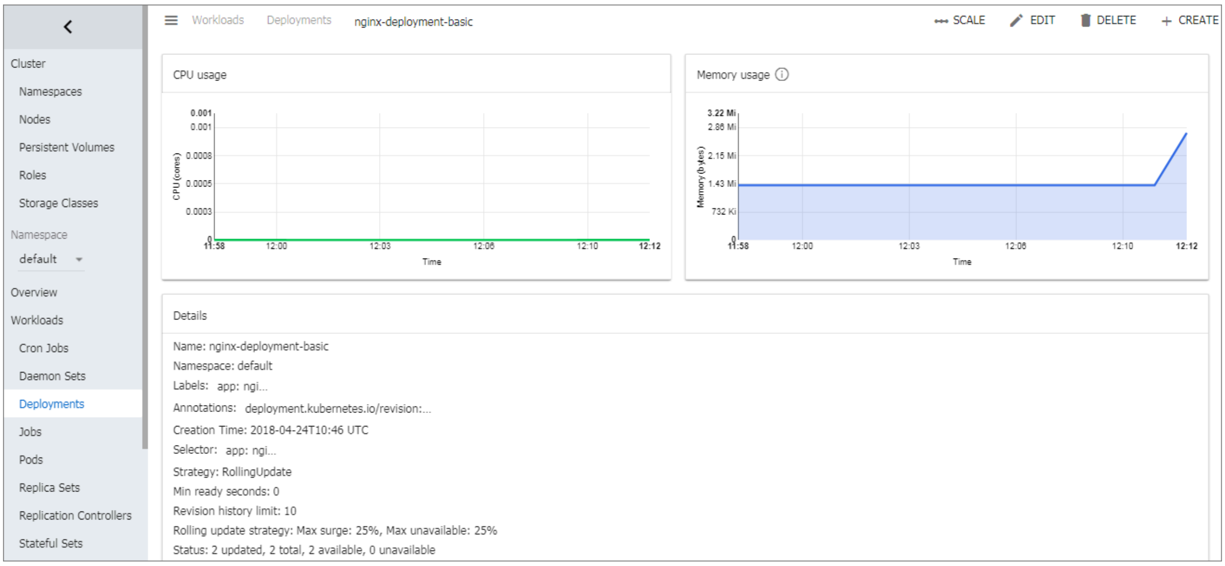


What's next

You can check the status of each Kubernetes object according to the icon on the left of the deployment list. indicates the object is being deployed. indicates the object has been deployed.

Additionally, you can click a deployment name to view the details of the running Web service. Specifically, you can view the replica sets included in the deployment, and the CPU usage and memory usage of these replica sets.

Note:
If no resources are displayed, we recommend that you wait a few minutes and then refresh the page.



4.14 Create a service

This topic describes how to create a Kubernetes service with Alibaba Cloud Container Service for Kubernetes (ACK).

Background information

A Kubernetes service, known as a service in this and related topics of Alibaba Cloud Container Service for Kubernetes, is an abstract object that defines a logical set of pods and a policy through which to access the pods. Usually, a label selector determines which set of pods are targeted by a service.

In a Kubernetes cluster, each pod has its own IP address, and the pods of a deployment can be removed at any time. However, this action changes the IP addresses of the pods. As a result, directly using IP addresses of pods is ineffective as the scenario does not provide high availability. By comparison, a Kubernetes service decouples the relationship between the frontend and the backend. Specifically, a Kubernetes service is a loose coupling service solution where the operations of the backend do not impact the frontend.

For more information, see Kubernetes service.

Limits

To create a service of the Server Load Balancer type, you can select an existing SLB instance for the service. Multiple services can reuse the same SLB instance. For reused SLB instances, the following limits apply:

- If an existing SLB instance is reused, a new listener is created for the SLB instance and the original listener of the SLB instance is overridden.
- Only the SLB instances that you manually create by using the console or API can be reused. If an SLB instance is automatically created by the system for a service, it cannot be reused by any other service. Otherwise, the reused SLB instance may be removed incidentally.
- The multiple services that reuse the same SLB instance cannot have the same frontend listening port. Otherwise, port conflicts will occur.
- The name of the listener and virtual server group of a reused SLB instance cannot be modified.
- An SLB instance cannot be reused by services across Kubernetes clusters.

Prerequisites

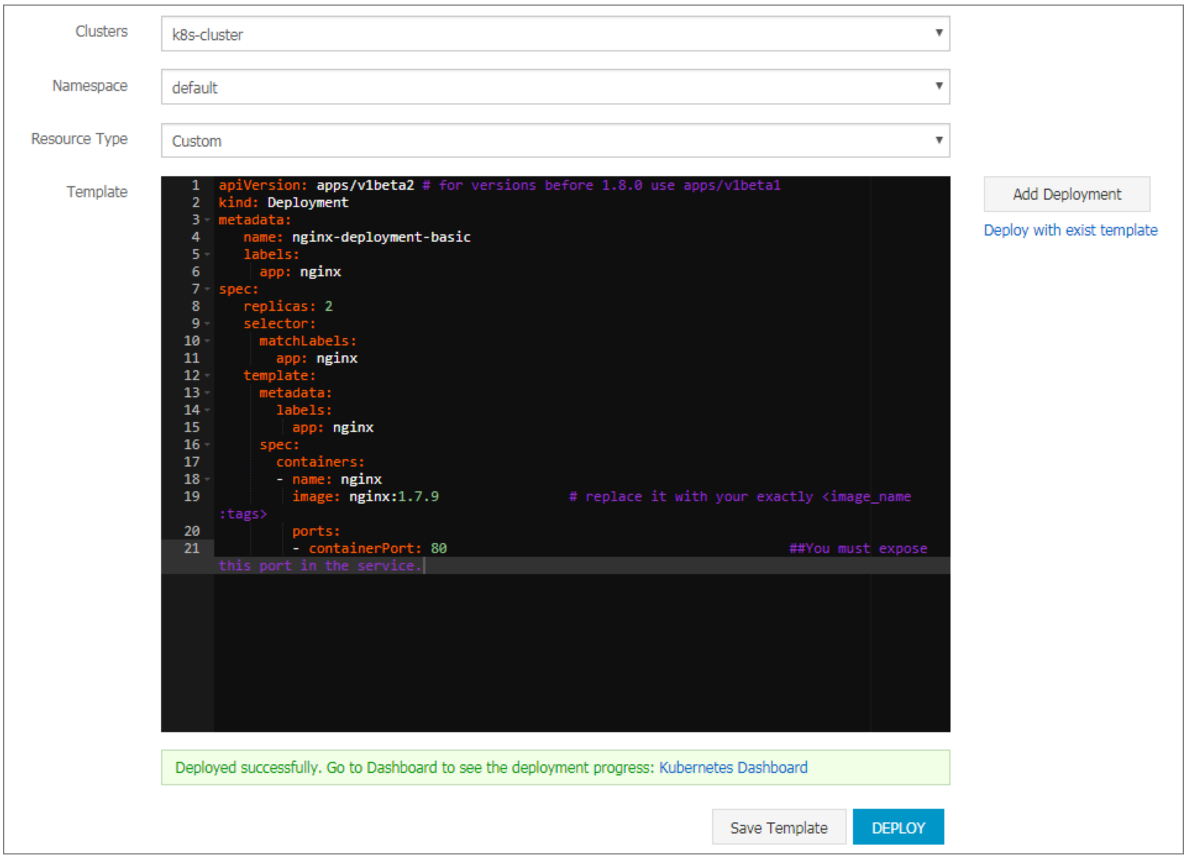
A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.

Step 1: Create a deployment

1. Log on to the Container Service console.

2. In the left-side navigation pane under Kubernetes, choose Applications > Deployments. Then click Create by Template in the upper-right corner.

3. Select the target cluster and namespace, and select a custom template or a sample template from the Resource Type drop-down list. Then, click DEPLOY.

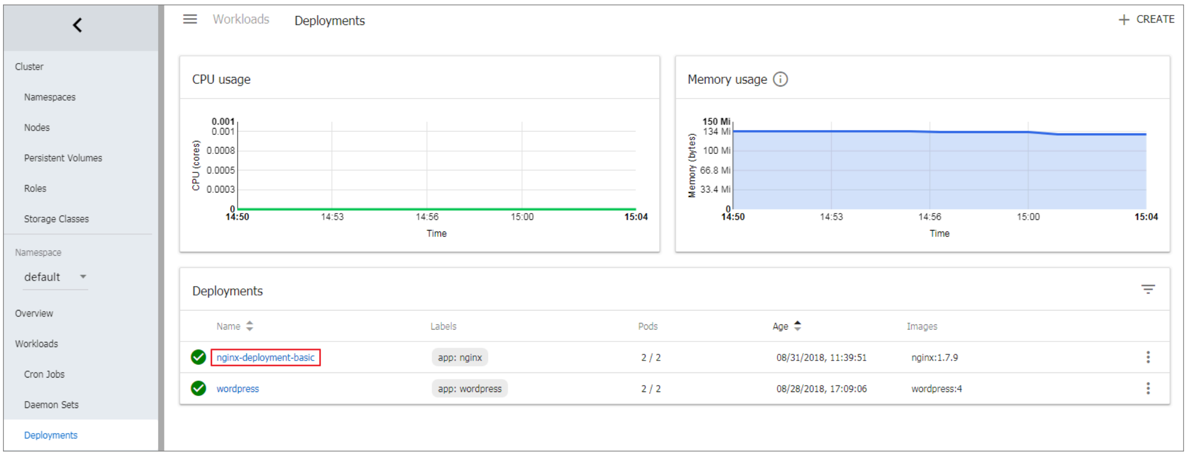


In this example, the sample template specifies an Nginx deployment.

```
apiVersion : apps / v1beta2 # for versions before 1 . 8 .
0 use apps / v1beta1
kind : Deployment
metadata :
   name : nginx - deployment - basic
   labels :
     app : nginx
spec :
   replicas : 2
   selector :
```

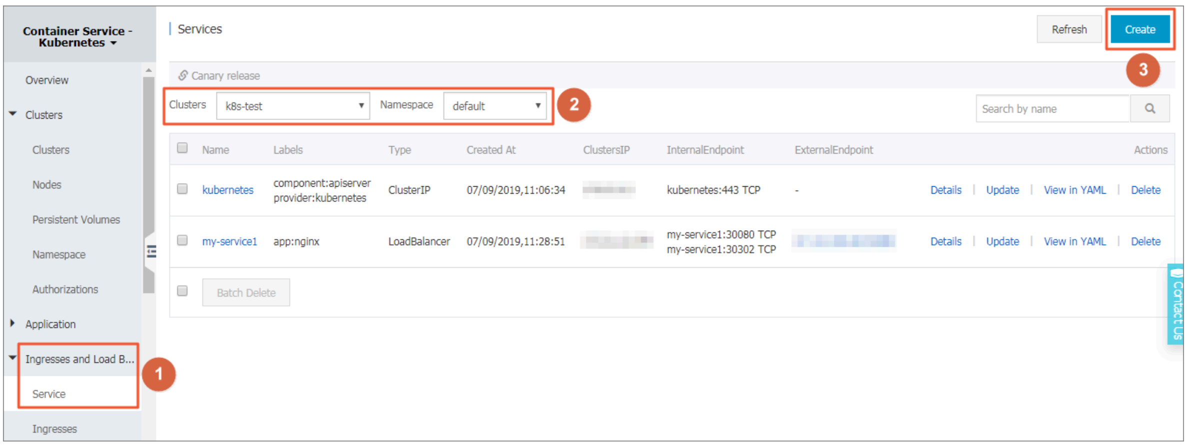
```
          matchLabel  s :
            app :  nginx
        template :
          metadata :
            labels :
              app :  nginx
          spec :
            containers :
           -  name :  nginx
              image :  nginx : 1 . 7 . 9                # replace  it
      with   your  exactly  < image_name : tags >
              ports :
             -  containerP ort :  80
            ## This   port   must   be   exposed   in   a   service .
```

4. Click Kubernetes Dashboard to view the running status of this deployment.

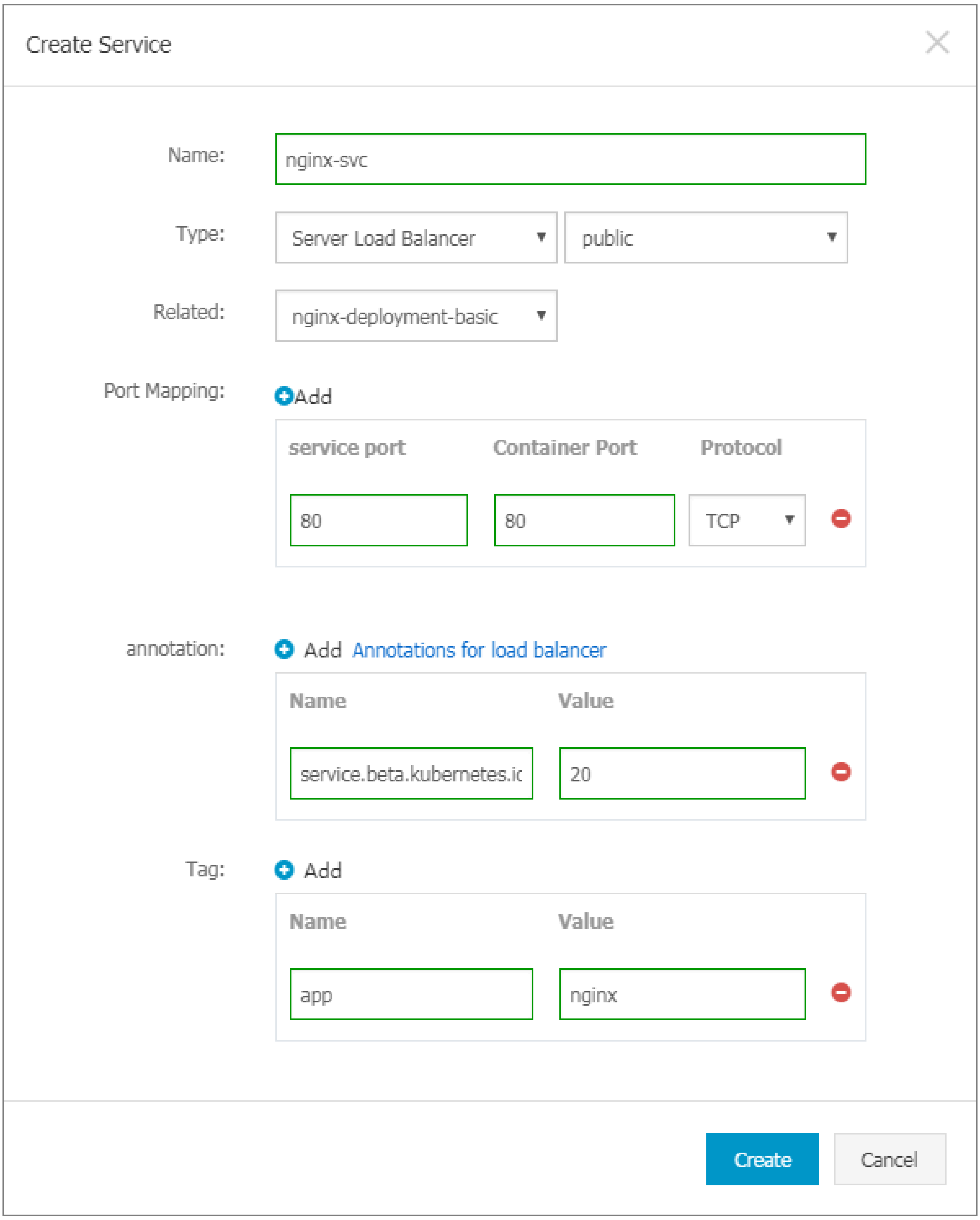


Step 2: Create a service

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Ingresses and Load Balancing > Service.
3. Select the target cluster and namespace. Then, click Create in the upper-right corner.



4. In the displayed dialog box, set service parameters.



- Name: Enter the service name. In this example, the service name is set to nginx-svc.
- Type: Select the service type, namely, the service access method.
  - Cluster IP: Exposes the service by using the internal IP address of your cluster. If you select this service type, the service is accessible only within the cluster. This is the default service type.

- Node port: Exposes the service by using the IP address and the static port (NodePort) of each node. A node port service routes to a cluster IP service that is automatically created. You can access the node port service from outside the cluster by requesting `< NodeIP >:< NodePort >`.
- Server Load Balancer: Alibaba Cloud Server Load Balancer (SLB) service. To create a service of this type, you can select an existing SLB instance for the service, or set the system to automatically create an SLB instance for the service.

  With this type of service, you can set an Internet or intranet access method for your application. An SLB instance can route to a node port service and a cluster IP service.

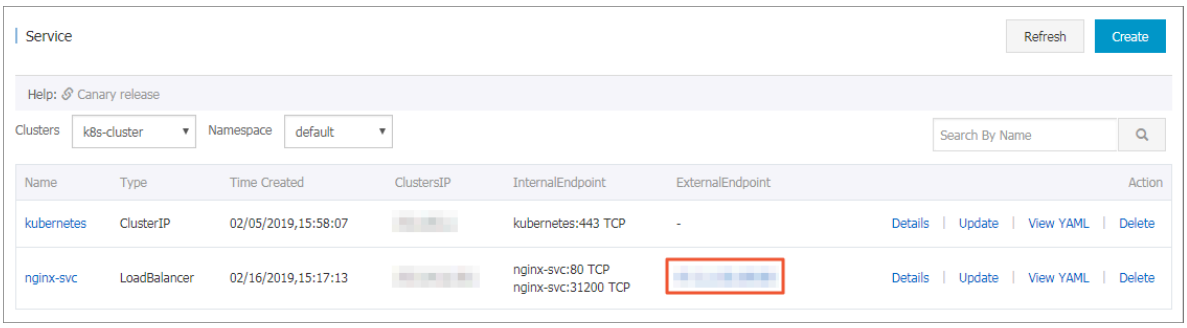
· Related: Select the backend object to associate with the service. In this example, the nginx-deployment-basic deployment created in the preceding step is associated with the service. If you do not associate the service with any objects, the system does not create any corresponding endpoint objects. In this case, you can manually associate the service with your own specific endpoints. For more information, see Services without selectors.

· Port Mapping: Add a service port number and a container port number. The container port number that you set must be the same as the port number of the container exposed by the pod.

· annotation: Add an annotation to the service. You can set SLB parameters. For example, to control the service traffic, you can set the peak bandwidth of the service to 20 Mbit/s by setting this parameter as `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20`. For more information, see Access services by using Server Load Balancer.

· Tag: Add a tag to the service to identify the service.

5. Click Create. The nginx-svc service is then displayed in the service list.



6. Enter the external endpoint of the nginx-svc service in your browser to access the service.

4.15 View a service

This topic describes how to view a Kubernetes service in Alibaba Cloud Container Service for Kubernetes.

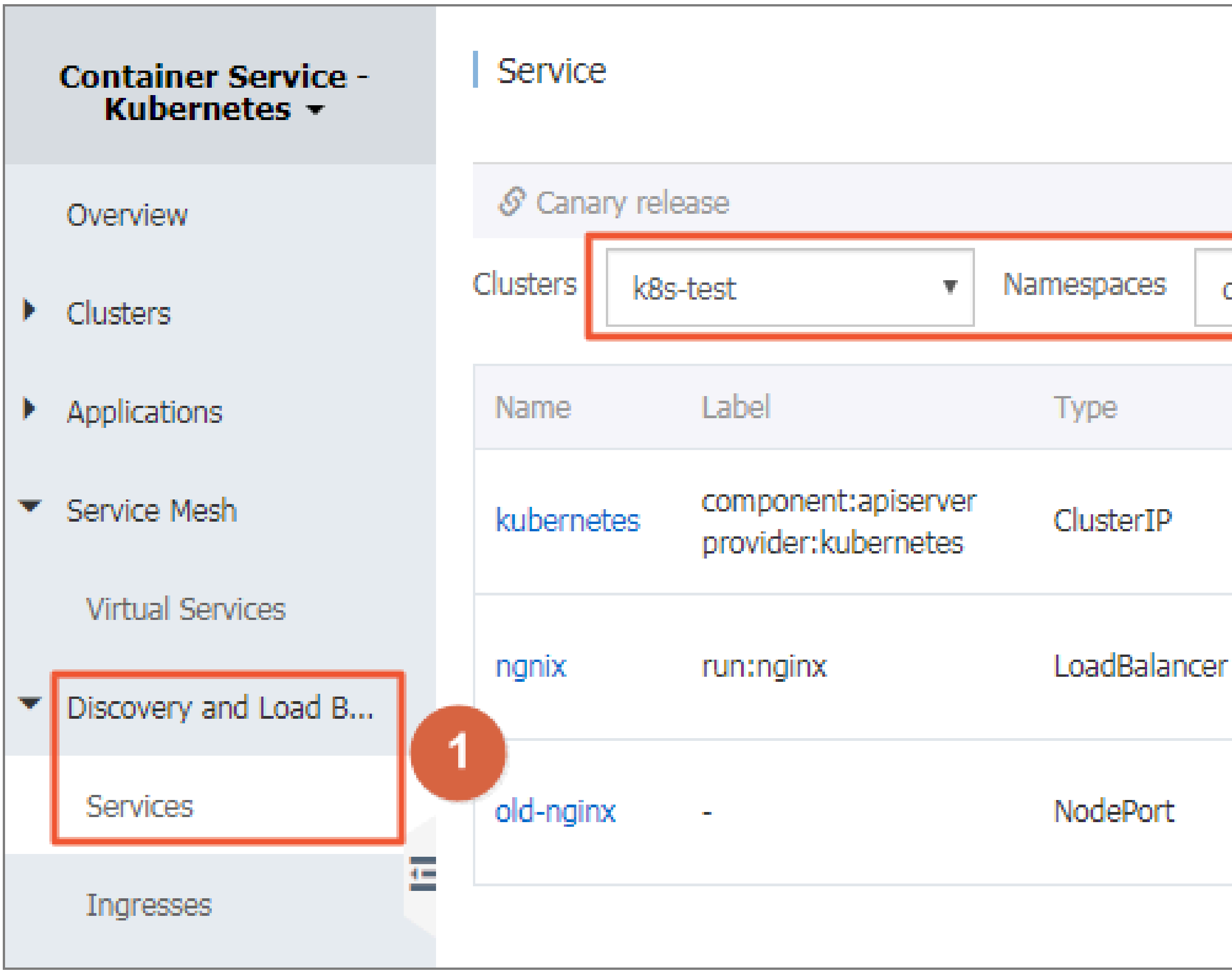
Context

If you set an external service when you create an application, the Kubernetes dashboard creates the external service, in addition to running containers. The service is used to pre-set a Server Load Balancer to distribute traffic to the containers.

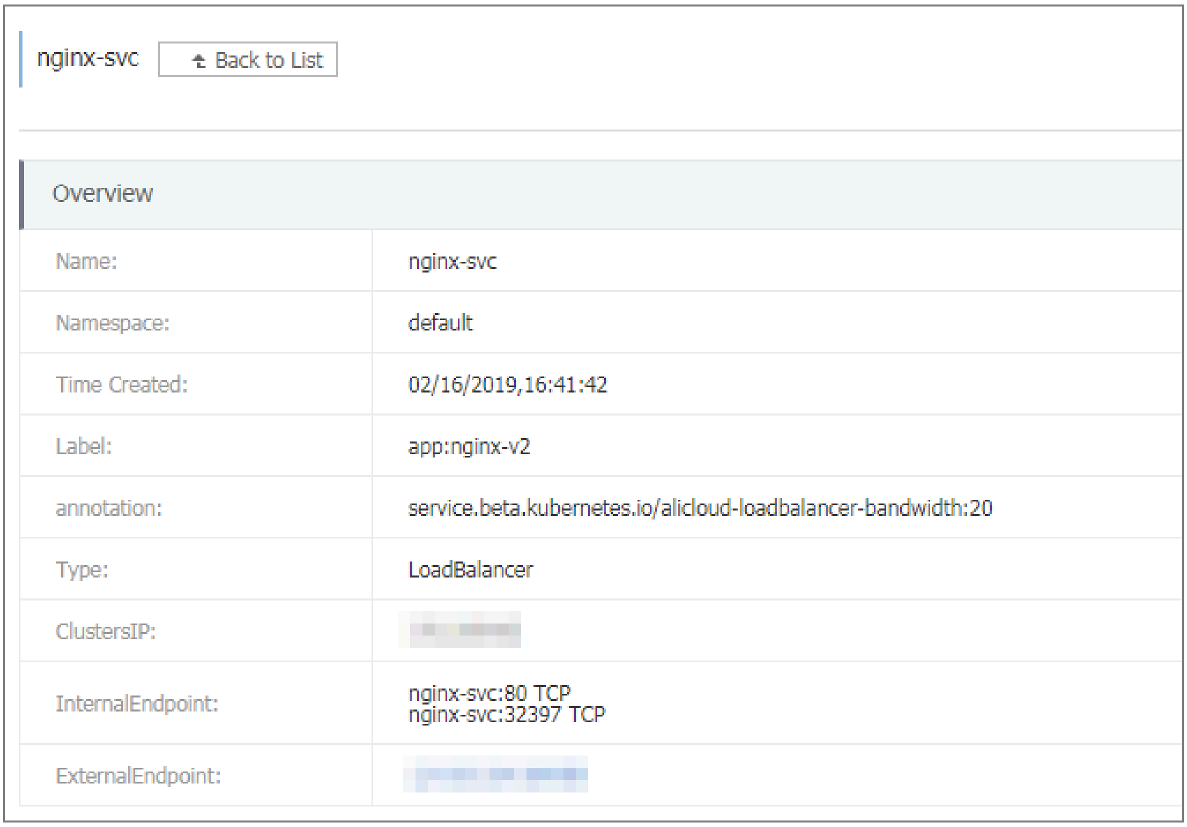
Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Discovery and Load Balancing > Services.

3. Select the target cluster and namespace, and then click Details on the right of the target service.



You can view the service name, service type, service creation time, cluster IP address, external endpoint, and other information. In this example, the external endpoint (IP address) assigned to the service is displayed. To access the Nginx application, click this IP address.



You can also open the Kubernetes dashboard of the target cluster and click Services in the left-side navigation pane to view all services of the cluster.

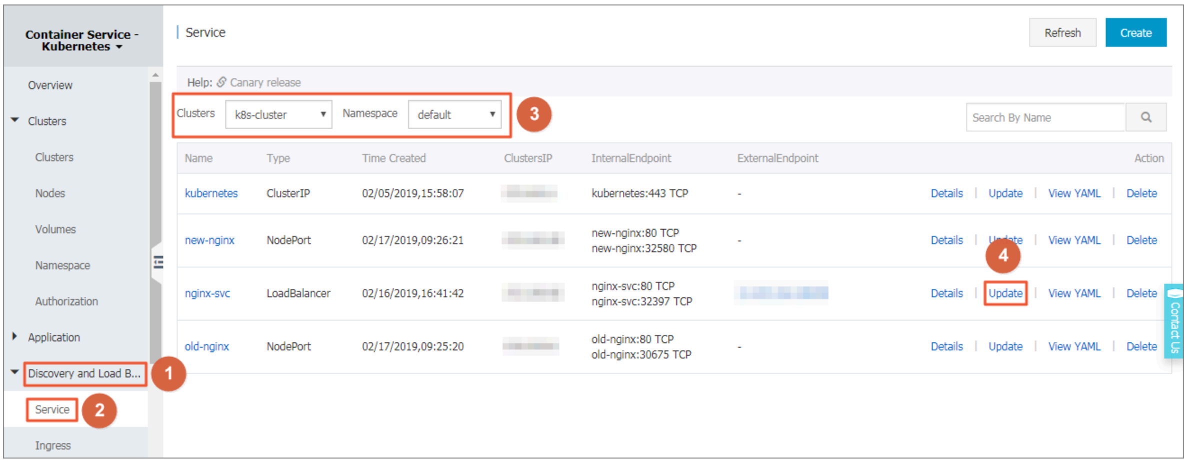
4.16 Update a service

This topic describes how to update a Kubernetes service in the Container Service console or the Kubernetes dashboard.

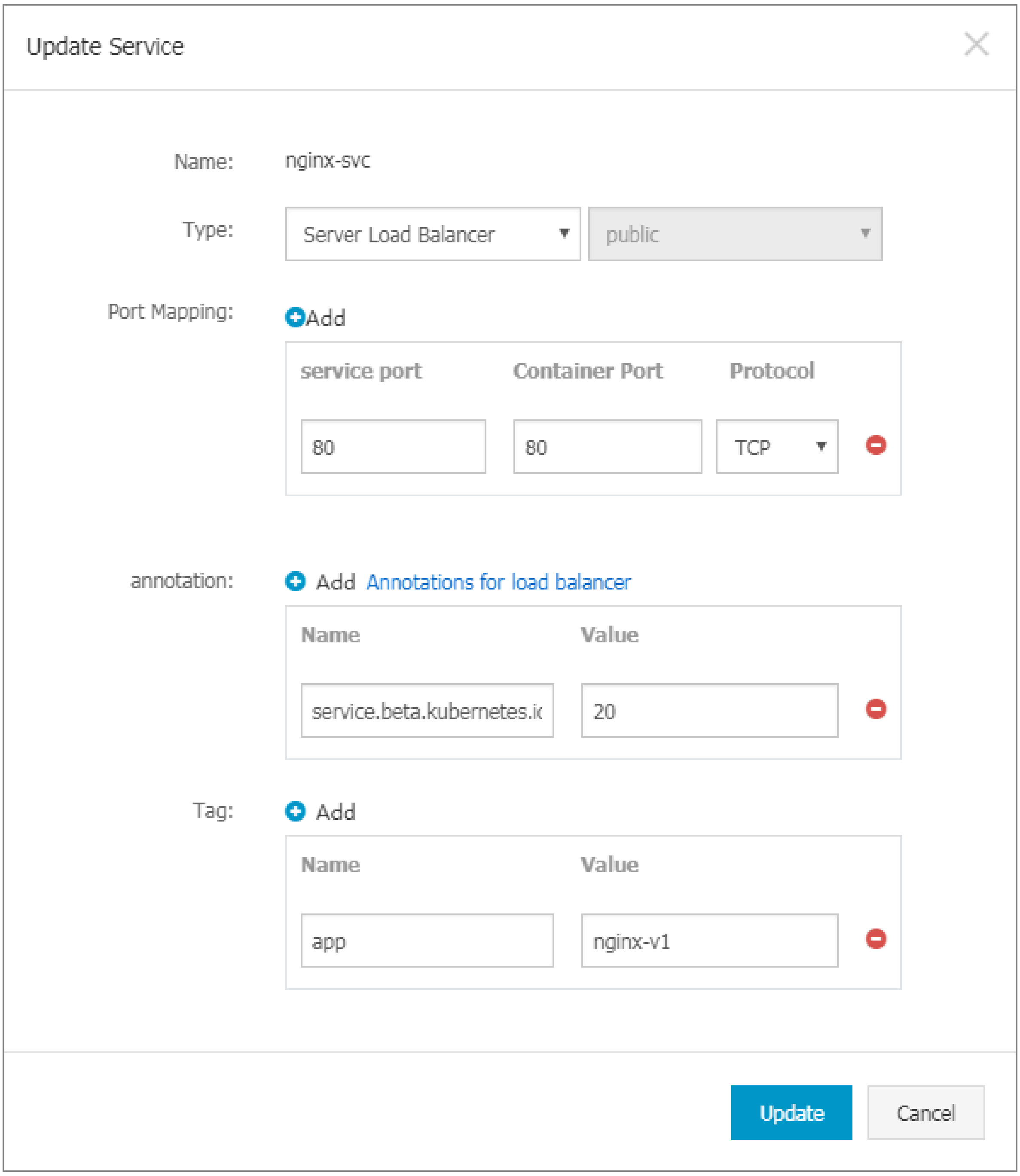
Update a service in the Container Service console

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Discovery and Load Balancing > Service.

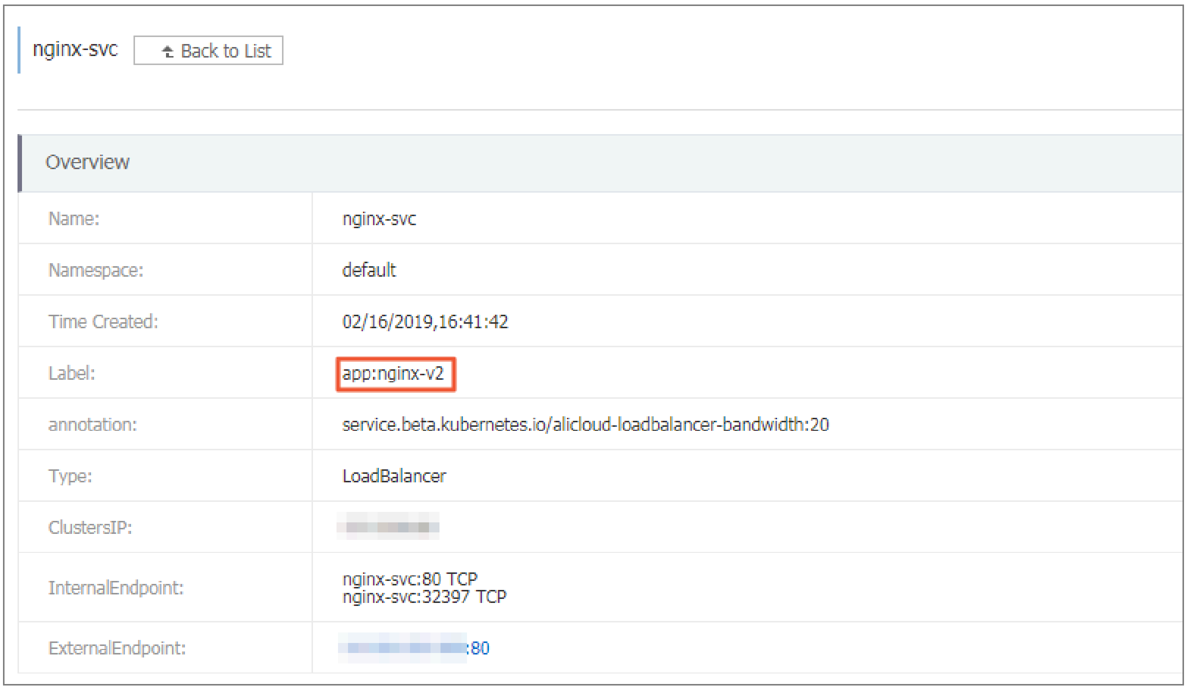
3. Select the target cluster and namespace, and then click Update on the right of the target service. In this example, the target service is named nginx-svc.



4. In the displayed dialog box, update the service parameters. Then, click Update. In this example, the service tag is updated from app:nginx-v1 to app:nginx-v2.

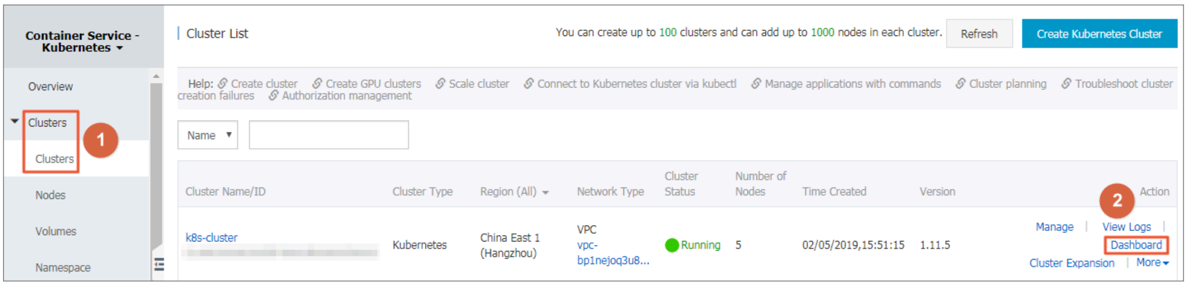


5. In the service list, click Details on the right of the target service to view the service updates. In this example, the updated service tag app:nginx-v2 is displayed.



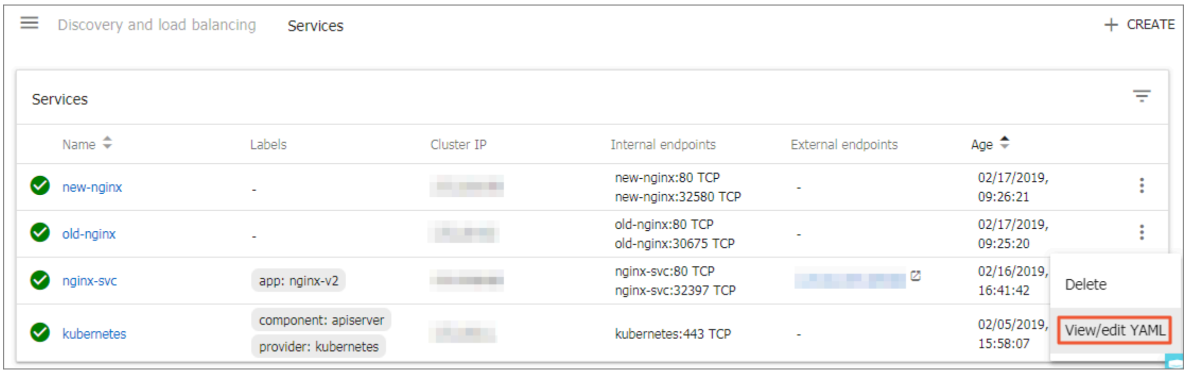
Update a service in the Kubernetes dashboard

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, click Clusters.
3. Click Dashboard on the right of the target cluster.

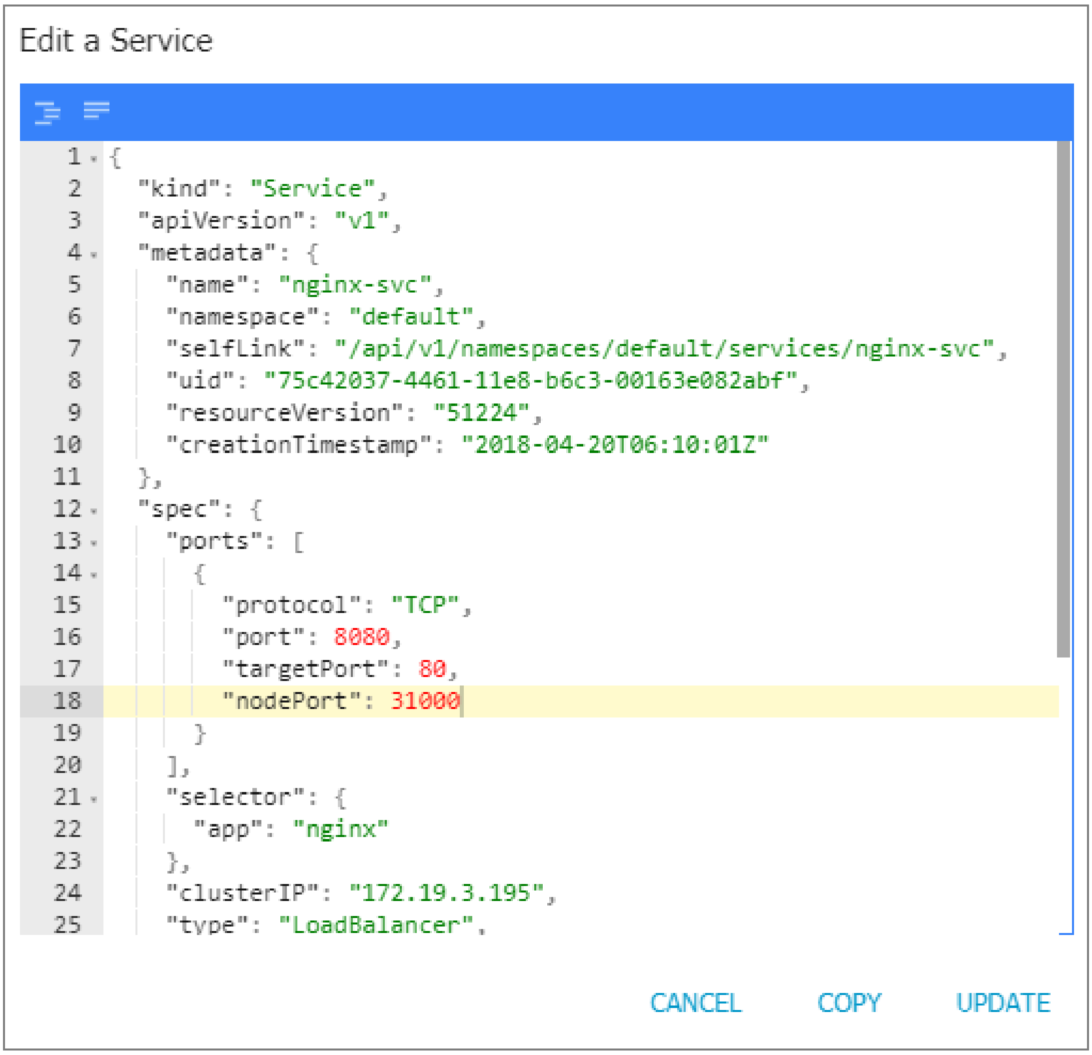


4. In the Kubernetes dashboard, select the target namespace and then click Services in the left-side navigation pane.

5. Click the icon on the right of the target service and then click View/edit YAML.



6. In the displayed dialog box, modify the service settings. Then, click UPDATE. In this example, the nodePort is changed to 31000.



4.17 Delete a service

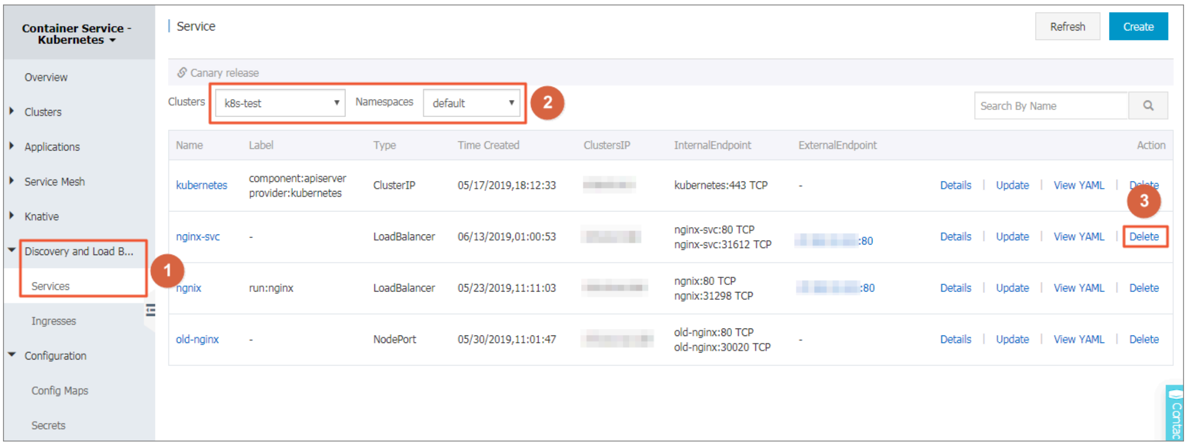
This topic describes how to delete a Kubernetes service in the Container Service console.

Prerequisites

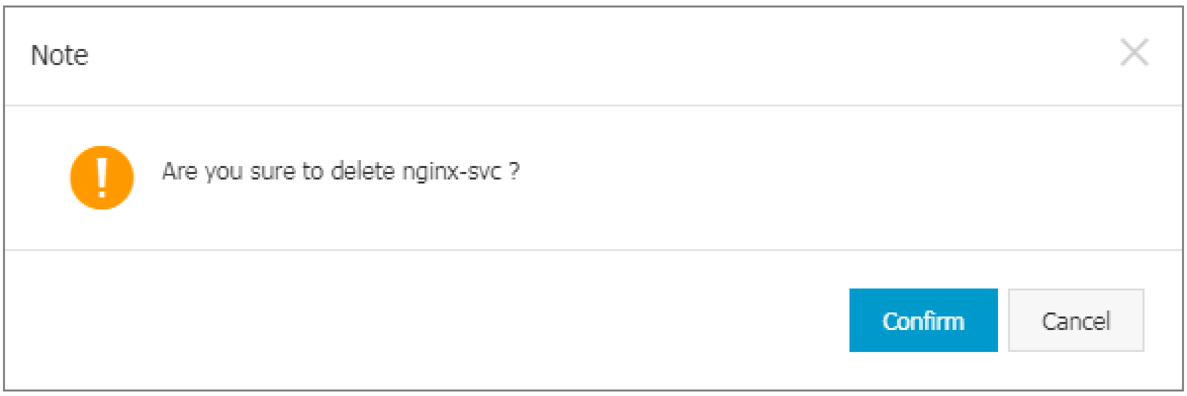
- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A service is created. For more information, see Create a service.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Discovery and Load Balancing > Services.
3. Select the target cluster and namespace, and then click Delete on the right of the target service. In this example, the target service is named nginx-svc.



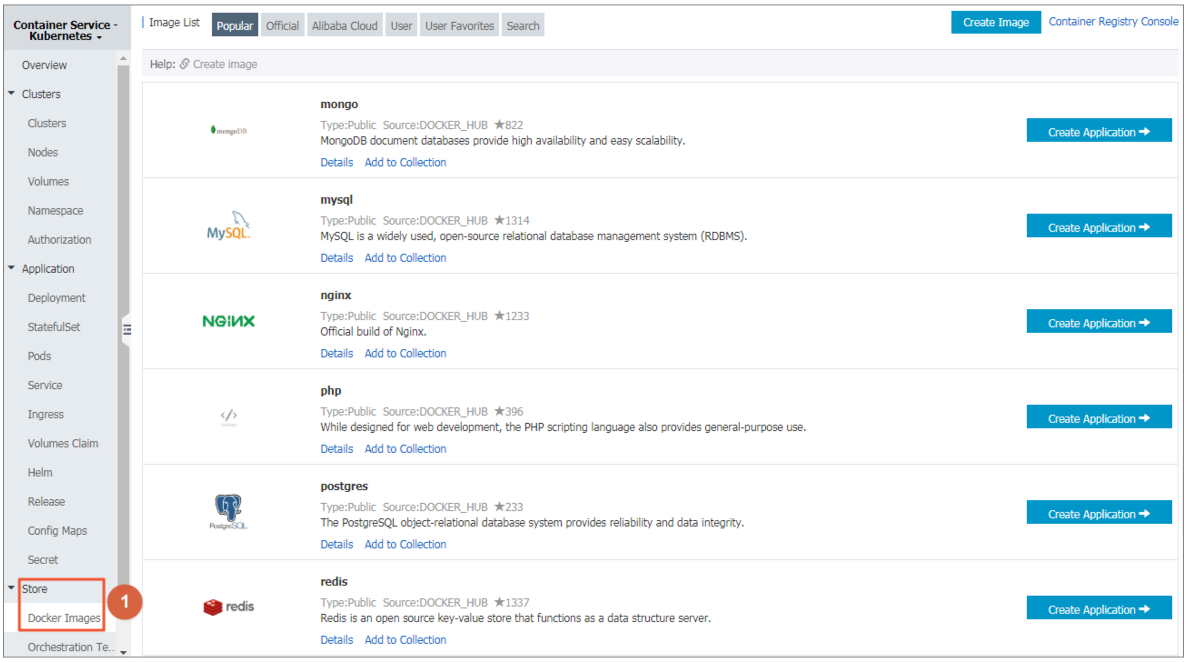
4. In the displayed dialog box, click Confirm.



4.18 View image list

Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Store > Docker Images in the left-side navigation pane.



You can view the image category.

- Popular: Some common images recommended by Container Service.
- Official: Official images provided by Docker Hub.

4.19 Use an image Secret

Container Service Kubernetes clusters support using image secrets through the web interface. You can create an image secret and use an existing image secret.

Prerequisites

- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have built a private image repository and uploaded your image to the repository. In this example, use Alibaba Cloud Container Registry. For more information, see Use a private image repository to create an application.

Context

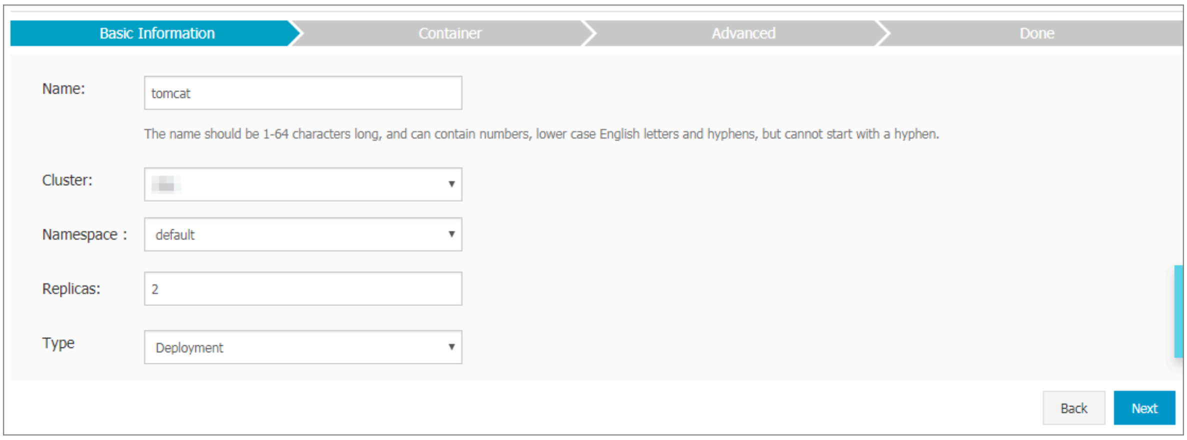
When you use a private image to create an application, you have to configure a secret for the image to secure the image. In the Container Service console, you can deliver the identity authentication information of the private image repository to Kubernetes through a secret of the docker-registry type.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments. Then, click Create by Image in the upper-right corner.
3. Configure Name, Cluster, Namespace, Replicas, and Type. The configured value of the replicas parameter specifies the number of pods contained in the application. Click Next.

   Note:
   In this example, select the Deployment type.

   If you do not configure Namespace, the system uses the default namespace by default.



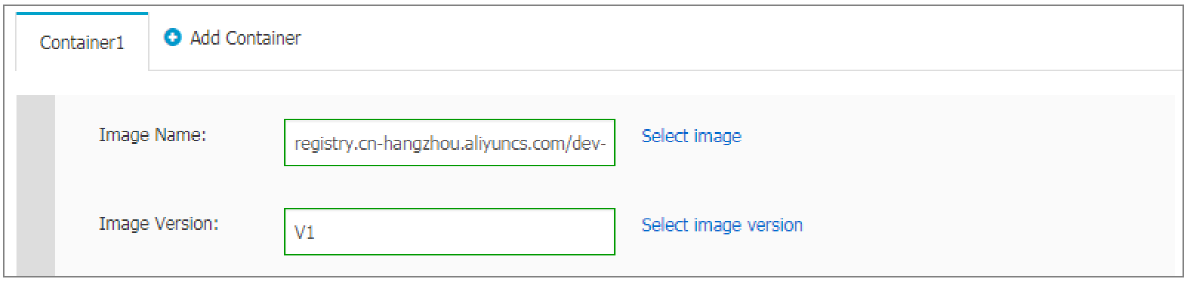
4. Configure containers.

   Note:
   This example describes only the configuration of the container image secret. For more information about container configuration, see Create a deployment application by using an image.

5. On the container configuration page, configure the image name first. Enter the private image address in the Image Name box. The format is `domainname / namespace / imagename`.

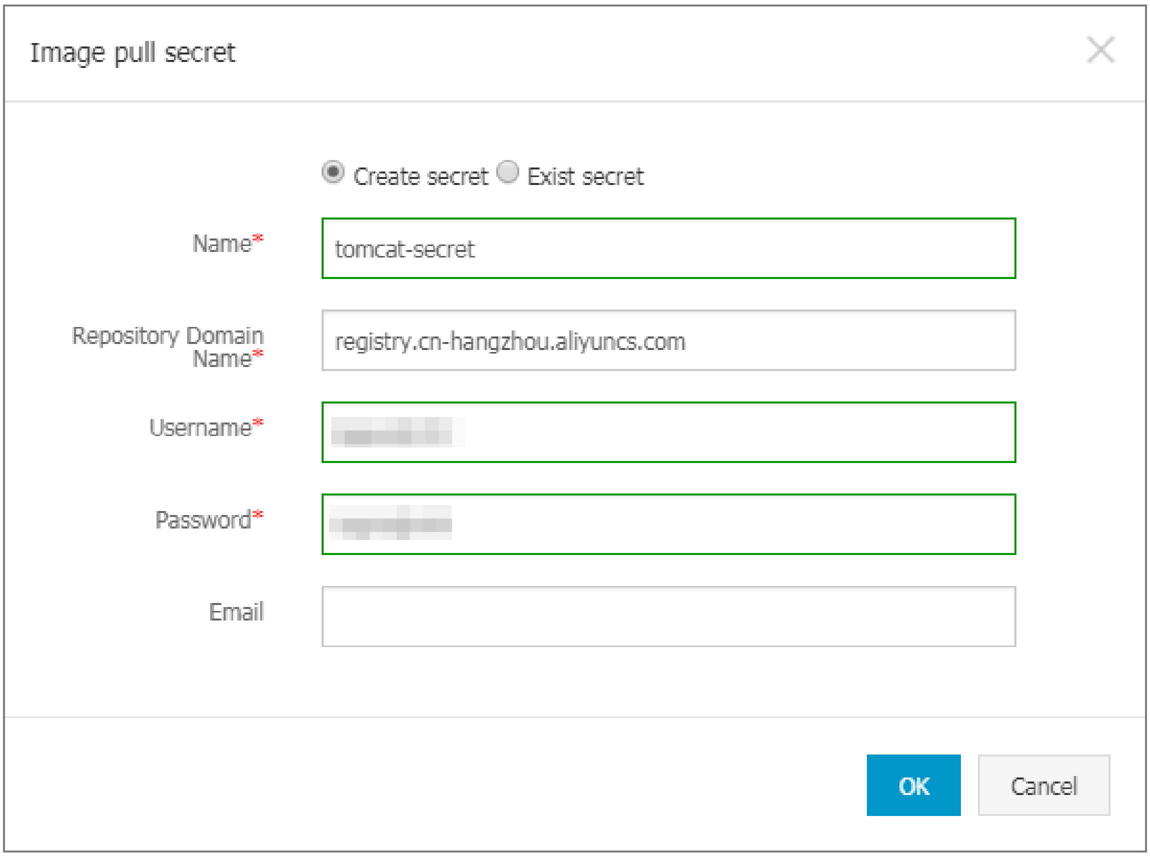
Note:
Public images do not require image secrets.

6. In the image version box, enter the private image address version.

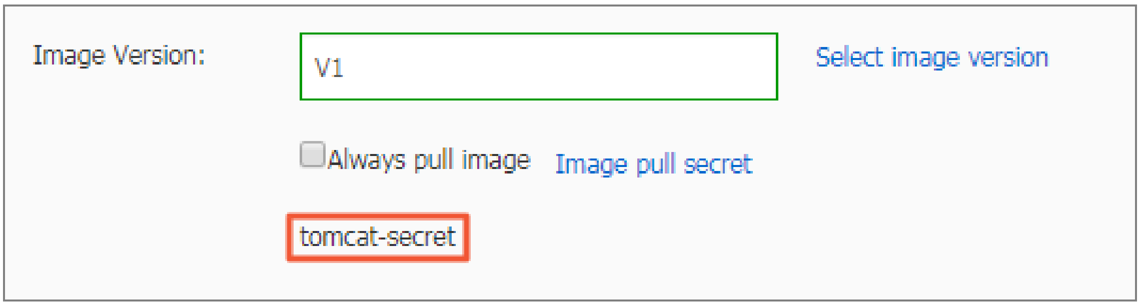


7. Click Image pull secret.

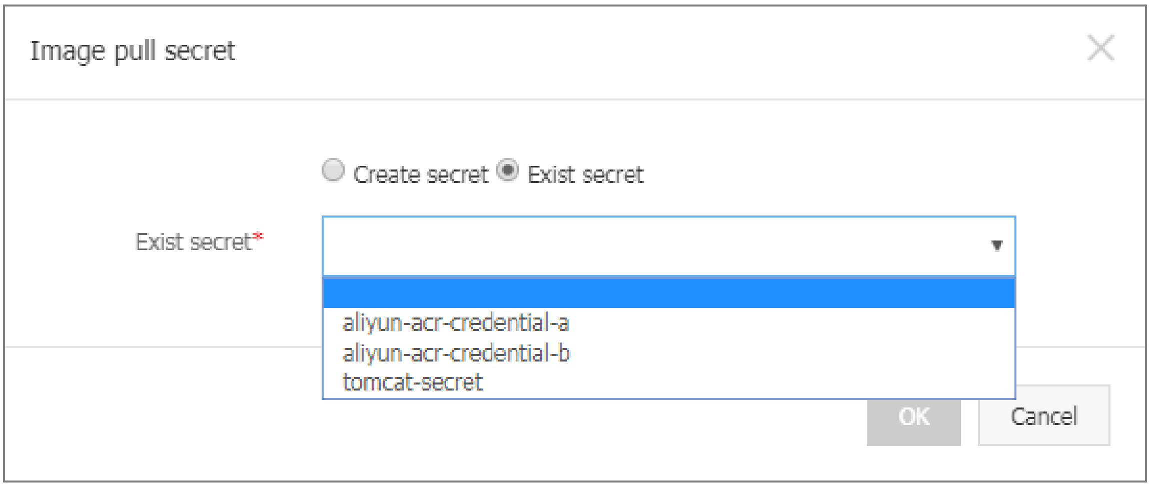
   - Select Create secret.
     - Name: Specifies the secret name. You can define it by yourself.
     - Repository Domain Name: Specified the Docker repository address. If you enter the Alibaba Cloud Container Service image repository in the image name box, the system automatically adds the repository address by default.
     - Username: Specifies the user name of the Docker repository. If you use Alibaba Cloud Container Registry, the username is your Alibaba Cloud account name.
     - Password: Specifies the logon password of the Docker repository. If you use Alibaba Cloud Container Registry, the password is the independent logon password for Container Registry.
     - Email: Specifies an email address. This is optional.



   Click OK. The created secrete is displayed on the page.



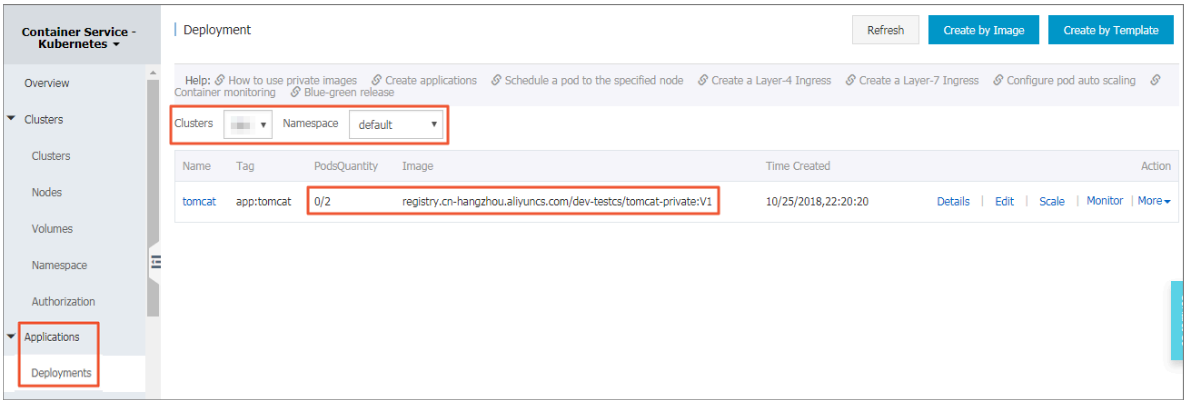
- You can also click Exist secret. You can pre-create a container image secret by using command lines or a YAML file. For information, see How to use private images in Kubernetes clusters and Use a private image repository to create an application.



8. After you complete the container configuration, click Next.
9. Follow the page guide to complete other configurations, and then click Create.
10. Click Applications > Deployments in the left-side navigation pane, and select the cluster and namespace in which the application is created to view the status of the tomcat application.

Note:

The system shows that the tomcat application runs properly, which indicates that you have used the tomcat private image through the secret.



4.20 Pull an image without a password

This topic describes how to pull a private image without a password from the Alibaba Cloud container image repository.

Prerequisites

You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.

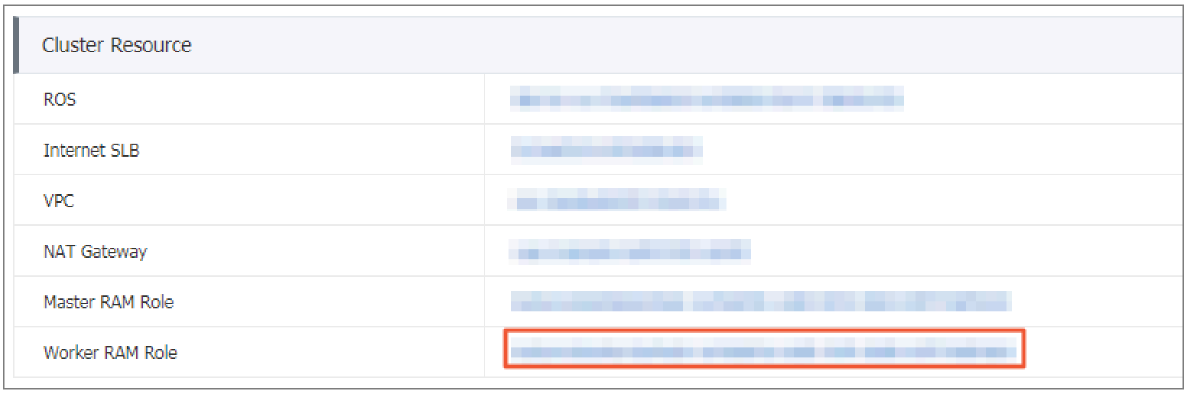
Context

- You can only pull a private image from an Alibaba Cloud container image repository that belongs to your account.
- You can pull a private image from a cross-region Alibaba Cloud container image repository.
- You can only perform this operation in multiple namespaces.
- You can pull a private image from an image repository of the Enterprise Edition of Alibaba Cloud Container Registry.
- Kubernetes clusters that support this function include:
  - Dedicated Kubernetes clusters
  - Managed Kubernetes clusters
  - Serverless Kubernetes clusters

- The following are Kubernetes cluster versions that support this function:
  - Dedicated Kubernetes cluster versions that are not earlier than v1.11.2 support this function by default. If the dedicated Kubernetes cluster version is earlier than v1.11.2, follow the procedures described in this topic.
  - All versions of managed Kubernetes clusters support this function.
  - All versions of serverless Kubernetes clusters support this function.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, click Clusters.
3. Click the target cluster name to view the cluster details.
4. In the Cluster Resources area, click Worker RAM Role.



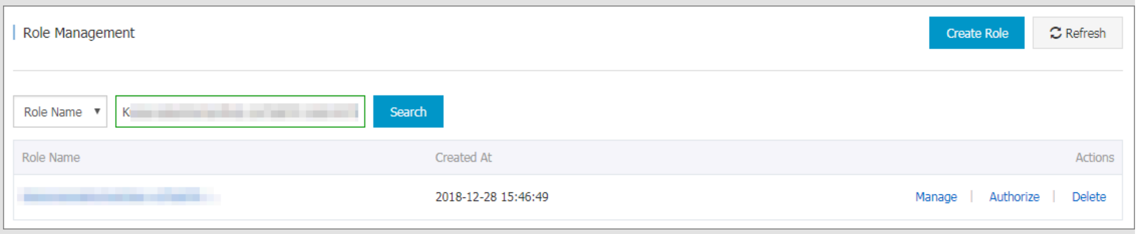
Note:

This topic uses the latest version of the RAM console.

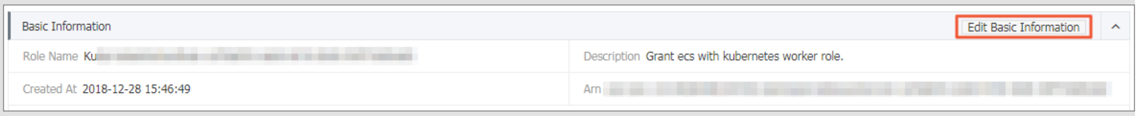
If you use an earlier version of the RAM console, you can modify the target policy document by using either of the following two methods:

Method 1

a. In the left-side navigation pane, click Roles, and then enter the Worker RAM Role name in the Role Name box. Click the target Role Name.

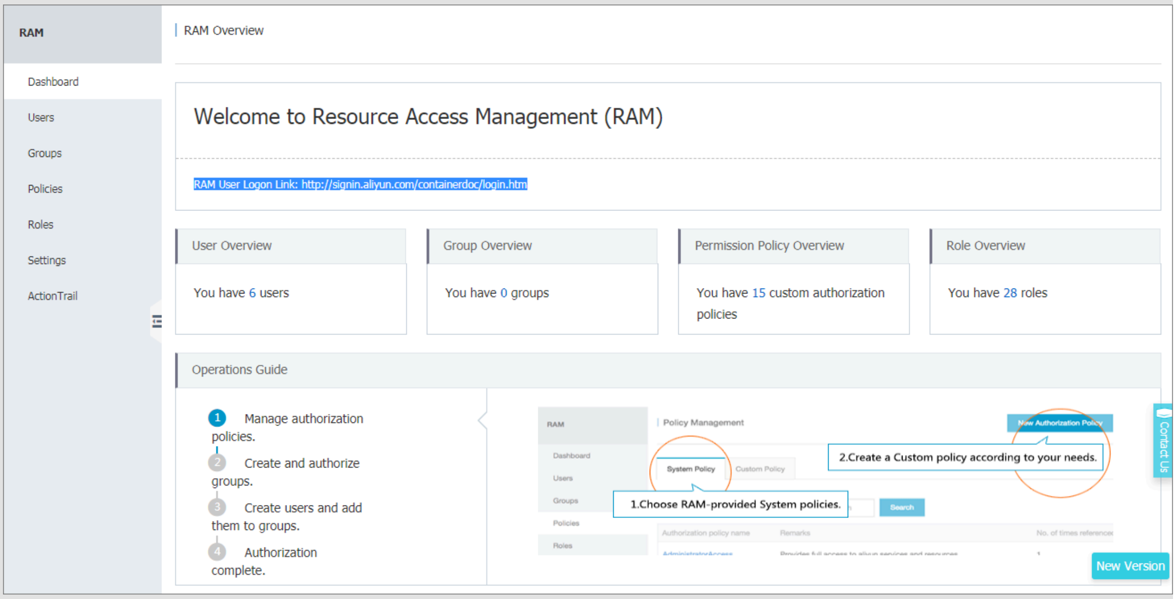


b. In the Basic Information area, click Edit Basic Information in the upper-right corner.

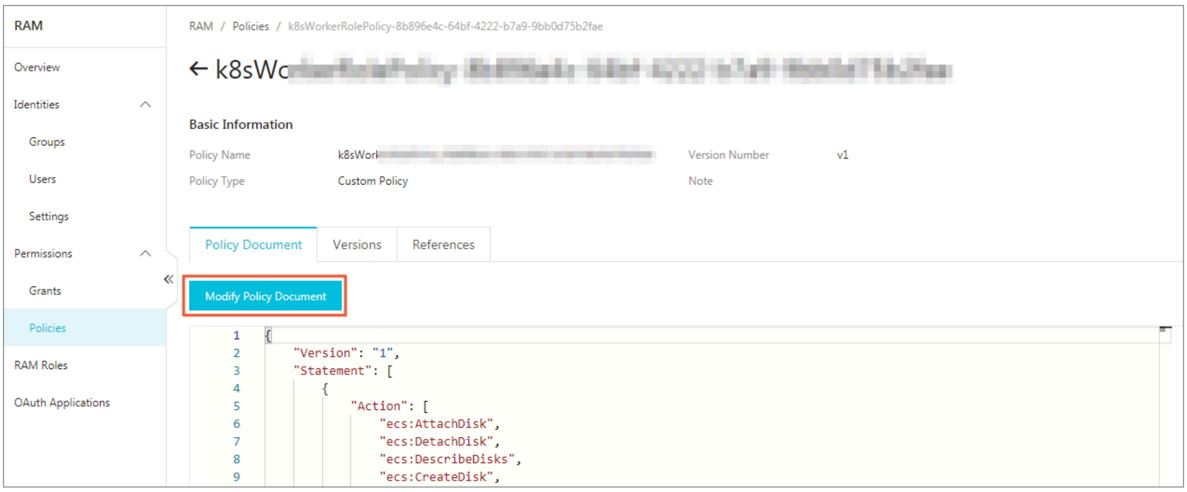


Method 2

In the lower-right corner of the RAM dashboard page, click New Version to switch to the latest version of the RAM console. In the Container Service console, click Worker RAM Role to log on to the RAM console.



5. On the RAM Roles page, click the policy name in the Permission area to view the policy details.
6. On the Policies page, click Modify Policy Document in the Policy Document area.



7. In the Policy Document area, add the following fields and then click OK.

```
{
            " Action ": [
              " cr : Get *",
              " cr : List *",
              " cr : PullReposi  tory "
            ],
            " Resource ": "*",
            " Effect ": " Allow "
         }
```

8. **Create the** *aliyun - acr - credential - helper* **service to refresh a temporary token of Container Registry.**

```
apiVersion :  v1
kind :  ConfigMap
metadata :
  name :  acr - configurat  ion
  namespace :  kube - system
data :
   # For   informatio  n   about   configurat  ion   descriptio  n
,  see   the   following   table .
    acr - api - version : " 2018 - 12 - 01 "
   # acr - registry : " xxx - registry .*. cr . aliyuncs . com ,
xxx - registry - vpc .*. cr . aliyuncs . com "
   # watch - namespace : " all "
   # expiring - threshold : " 15m "
---
apiVersion :  v1
kind :  ServiceAcc  ount
metadata :
  name :  aliyun - acr - credential - helper
  namespace :  kube - system
---
apiVersion :  rbac . authorizat  ion . k8s . io / v1beta1
kind :  ClusterRol  e
metadata :
  name :  aliyun - acr - credential - helper
rules :
 -  apiGroups :
     - ""
    resources :
     -  namespaces
     -  configmaps
    verbs :
     -  get
     -  list
     -  watch
 -  apiGroups :
     - ""
    resources :
     -  serviceacc  ounts
     -  secrets
    verbs :
     -  create
     -  update
     -  patch
     -  get
     -  list
     -  watch
---
apiVersion :  rbac . authorizat  ion . k8s . io / v1beta1
kind :  ClusterRol  eBinding
metadata :
  name :  aliyun - acr - credential - helper
roleRef :
  apiGroup :  rbac . authorizat  ion . k8s . io
  kind :  ClusterRol  e
  name :  aliyun - acr - credential - helper
subjects :
 -  kind :  ServiceAcc  ount
    name :  aliyun - acr - credential - helper
    namespace :  kube - system
```

```
---
 apiVersion :  apps / v1beta2
 kind :  Deployment
 metadata :
   name :  aliyun - acr - credential - helper
   namespace :  kube - system
   annotation  s :
     component . version : " v19 . 01 . 28 "
     component . revision : " v1 "
   labels :
     app :  aliyun - acr - credential - helper
 spec :
   replicas :  1
   selector :
     matchLabel  s :
       app :  aliyun - acr - credential - helper
   template :
     metadata :
       labels :
         app :  aliyun - acr - credential - helper
     spec :
       serviceAcc  ountName :  aliyun - acr - credential - helper
       containers :
      - name :  aliyun - acr - credential - helper
         image :  registry . cn - hangzhou . aliyuncs . com / acs /
 aliyun - acr - credential - helper : v19 . 01 . 28 . 0 - f063330 -
 aliyun
         imagePullP  olicy :  Always
         env :
          - name :  POD_NAME
            valueFrom :
              fieldRef :
                fieldPath :  metadata . name
          - name :  POD_NAMESP  ACE
            valueFrom :
              fieldRef :
                fieldPath :  metadata . namespace
         volumeMoun  ts :
        - name :  localtime
          mountPath : / etc / localtime
          readOnly :  true
       volumes :
        - name :  localtime
          hostPath :
            path : / etc / localtime
            type :  File
       nodeSelect  or :
         beta . kubernetes . io / os :  linux
```

Note:

The component Credential Helper of Alibaba Cloud Container Registry (ACR) is set with a ConfigMap. The settings of this component automatically take effect.

Table 4-1: ACR Credential Helper

Setting	Description	Default value
`acr - api - version`	The version of the ACR API. Note: The API of the Enterprise Edition of Alibaba Cloud Container Registry is supported.	2018-12-01
`acr - registry`	The target registry from which you pull an image without a password.	registry.*.aliyuncs.com,registry-vpc.*.aliyuncs.com,xxx-registry.*.cr.aliyuncs.com,xxx-registry-vpc.*.cr.aliyuncs.com Note: To set multiple registries, use commas (,) to separate them.
`watch - namespace`	The namespace that you use to pull an image without a password.	default Note: If you set this parameter as `All` , then all namespaces can be used to pull an image without a password. To set multiple namespaces, use commas (,) to separate them.

Setting	Description	Default value
`expiring - threshold`	The valid period before a temporary token in your local cache expires. **Note:** A temporary token of Container Registry is located in your local cache and is automatically refreshed at intervals of this valid period.	15m

5 Workflow

5.1 Create a workflow

This topic describes how to create a workflow by using the Container Service console or Ags CLI.

Background information

Based on Argo, the workflows developed by Alibaba Cloud provide containerized workflows for Alibaba Cloud Container Service for Kubernetes. Specifically, a workflow is implemented as a Kubernetes Custom Resource Definition (CRD). As such, you can use kubectl to manage workflows, and integrate them with other Kubernetes services, such as volumes, Secrets, and Role-Based Access Control (RBAC). At the backend, the workflow controller provides complete workflow features such as parameter substitution, artifacts, fixtures, loops, and recursive workflows.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- The Master node of the Kubernetes cluster is accessible. For more information, see Connect to a Kubernetes cluster by using kubectl.

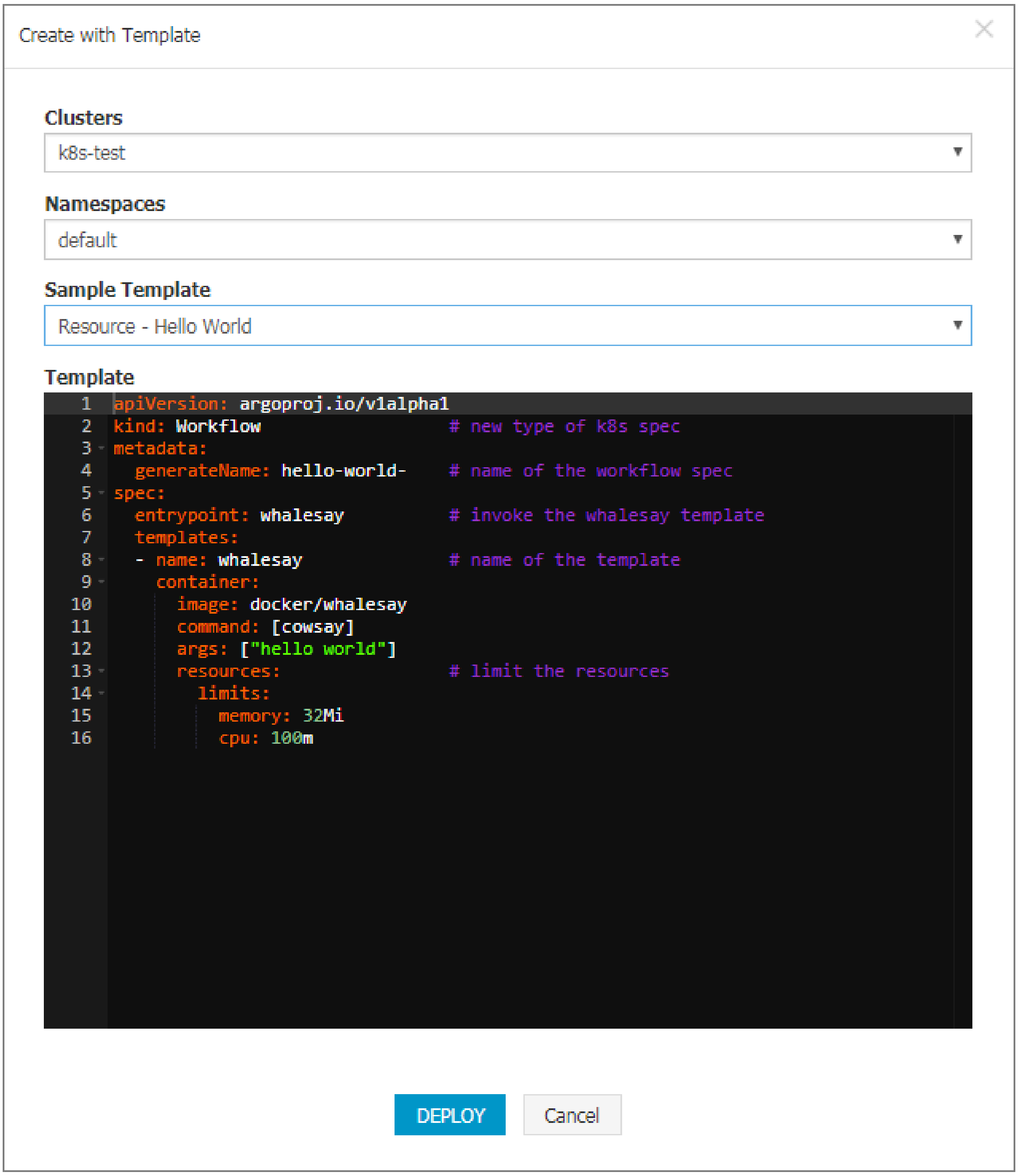
Procedure

- Use the Container Service console to create a workflow named Hello World

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Workflow.
3. In the upper-right corner, click Create by Template.
4. Set the template parameters.
   - Clusters: Select the target cluster.
   - Namespaces: Select the target namespace. The default namespace is used.
   - Sample Template: Select a sample YAML template or customize a YAML template.

   Note:

Alibaba Cloud Container Service for Kubernetes offers you with the sample YAML templates of various resources.

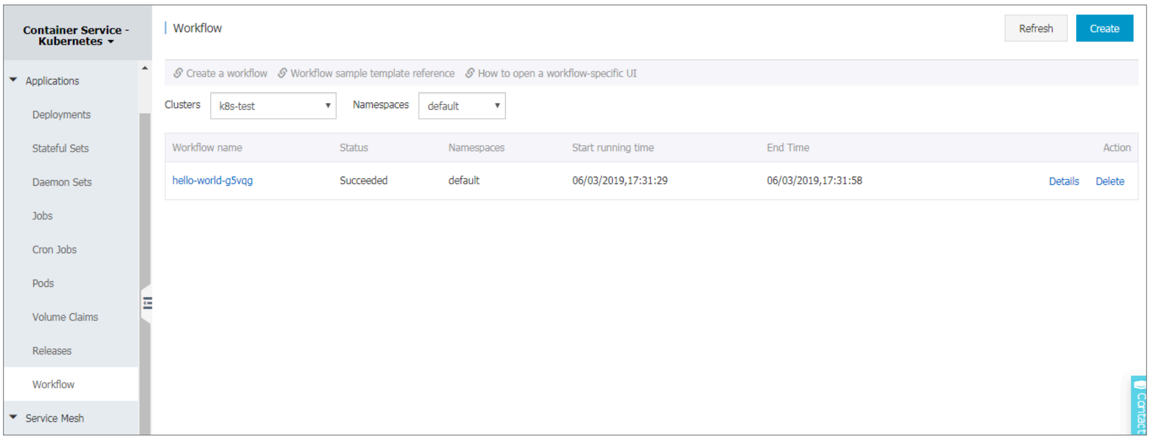


The following is a sample YAML template of the workflow named Hello World:

```
apiVersion : argoproj . io / v1alpha1
kind : Workflow                          # new type of k8s
spec
metadata :
  generateNa me : hello - world -      # name of the
workflow spec
spec :
  entrypoint : whalesay                 # invoke the whalesay
template
  templates :
 - name : whalesay                      # name of the template
```

```
      container :
        image :  docker / whalesay
        command : [ cowsay ]
        args : [" hello   world "]
        resources :                  #  limit   the   resources
          limits :
            memory :  32Mi
            cpu :  100m
```

5. Click DEPLOY.
6. On the Workflow page, find the target workflow. Then, in the Action column, click Details to view the overview and container group information of the workflow.



- Use the command line interface (CLI) to create a workflow named Parameters

Note:

Ags CLI is a CLI tool customized by Alibaba Cloud. This tool is compatible to Argo. With Ags CLI, you can submit, check, modify, and delete workflows. For more information, see Introduction to AGS CLI.

1. Create the file *arguments - parameters . yaml* , and then copy the following code to the file:

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  hello - world - parameters -
spec :
 #  invoke   the   whalesay   template   with
 # " hello   world "  as   the   argument
 #  to   the   message   parameter
  entrypoint :  whalesay
  arguments :
    parameters :
   -  name :  message
      value :  hello   world

  templates :
 -  name :  whalesay
```

```
      inputs :
        parameters :
       - name : message         # parameter  declaratio  n
      container :
       # run  cowsay  with  that  message  input  parameter
    as  args
        image : docker / whalesay
        command : [ cowsay ]
        args : ["{{ inputs . parameters . message }}"]
```

2. Run the `ags submit arguments - parameters . yaml - p message =" goodbye world "` command.

You can create more workflows by modifying the sample workflow templates. For more information, see Sample workflow templates.

5.2 Sample workflow templates

This topic provides several sample workflow templates that can be used to create workflows.

Steps

This sample workflow template can be used to create multi-step workflows, define more than one template in a workflow specification, and create nested workflows.

Note:

We recommend that you read the comments to ensure qualified code.

```
apiVersion : argoproj . io / v1alpha1
kind : Workflow
metadata :
  generateNa me : steps -
spec :
  entrypoint : hello - hello - hello

 # This  spec  contains  two  templates : hello - hello - hello
  and  whalesay
  templates :
 - name : hello - hello - hello
   # Instead  of  just  running  a  container
   # This  template  has  a  sequence  of  steps
    steps :
   - - name : hello1             # hello1  is  run  before
the  following  steps
        template : whalesay
        arguments :
          parameters :
         - name : message
           value : " hello1 "
   - - name : hello2a            # double  dash => run  after
  previous  step
        template : whalesay
```

```
        arguments :
          parameters :
         -  name :  message
            value : " hello2a "
      -  name :  hello2b            #  single   dash  =>  run   in
parallel   with   previous   step
        template :  whalesay
        arguments :
          parameters :
         -  name :  message
            value : " hello2b "

 #  This   is   the   same   template   as   from   the   previous
example
 -  name :  whalesay
    inputs :
      parameters :
     -  name :  message
    container :
      image :  docker / whalesay
      command : [ cowsay ]
      args : ["{{ inputs . parameters . message }}"]
```

The preceding workflow prints a hello-hello-hello template that contains three distinct hello steps. The first step named `hello1` runs in sequence, whereas the next two steps named `hello2a` and `hello2b` run parallel with each other. By using the Ags CLI command, you can display the running records of this workflow specification through the following tree-structure diagram:

```
STEP                                       PODNAME
#  arguments - parameters - rbm92
├---#  hello1                      steps - rbm92 - 2023062412
└-·-#  hello2a                     steps - rbm92 - 685171357
  └-#  hello2b                     steps - rbm92 - 634838500
```

Directed acyclic graph (DAG)

This sample workflow template can be also used to specify the sequence of steps in a workflow. You can define the workflow as a directed acyclic graph (DAG) by specifying the dependencies of each task. This method can help to simplify complex workflows by allowing a maximum number of tasks to be run in parallel.

The following workflow template shows the sequence of steps as follows:

1. Step A runs first having no dependency.
2. When step A is complete, steps B and C run in parallel.
3. When both B and C are complete, step D runs.

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  dag - diamond -
spec :
```

```
   entrypoint :  diamond
   templates :
  -  name :  echo
     inputs :
       parameters :
      -  name :  message
     container :
       image :  alpine : 3 . 7
       command : [ echo , "{{ inputs . parameters . message }}"]
  -  name :  diamond
     dag :
       tasks :
      -  name :  A
         template :  echo
         arguments :
           parameters : [{ name :  message ,  value :  A }]
      -  name :  B
         dependenci  es : [ A ]
         template :  echo
         arguments :
           parameters : [{ name :  message ,  value :  B }]
      -  name :  C
         dependenci  es : [ A ]
         template :  echo
         arguments :
           parameters : [{ name :  message ,  value :  C }]
      -  name :  D
         dependenci  es : [ B ,  C ]
         template :  echo
         arguments :
           parameters : [{ name :  message ,  value :  D }]
```

A dependency graph may have multiple roots. The templates called from a DAG or Steps template can be a DAG or Steps template. This allows you to separate a complex workflow into manageable parts.

Secrets

This sample workflow templates supports the same secret syntax and mechanisms as the Kubernetes pod specification. Through using this template, you can access a secret that functions as an environment variable or volume.

```
#  To   run   this   example ,  first   create   the   secret   by
 running :
#  kubectl   create   secret   generic   my - secret  -- from -
 literal = mypassword = S00perS3cr  etPa55word
 apiVersion :  argoproj . io / v1alpha1
 kind :  Workflow
 metadata :
   generateNa  me :  secret - example -
 spec :
   entrypoint :  whalesay
  #  To   access   secrets   as   files ,  add   a   volume   entry
 in   spec . volumes []  and
  #  then   in   the   container   template   spec ,  add   a   mount
   using   volumeMoun  ts .
   volumes :
  -  name :  my - secret - vol
     secret :
```

```
      secretName :  my - secret      #  name   of   an   existing
k8s   secret
  templates :
 -  name :  whalesay
    container :
      image :  alpine : 3 . 7
      command : [ sh , - c ]
      args : ['
        echo  " secret   from   env : $ MYSECRETPA  SSWORD ";
        echo  " secret   from   file : ` cat  / secret / mountpath /
mypassword `"
     ']
     #  To   access   secrets   as   environmen  t   variables ,  use
  the   k8s   valueFrom   and
     #  secretKeyR  ef   constructs .
      env :
     -  name :  MYSECRETPA  SSWORD   #  name   of   env   var
        valueFrom :
          secretKeyR  ef :
            name :  my - secret      #  name   of   an   existing
k8s   secret
            key :  mypassword      # ' key '  subcompone  nt   of
the   secret
      volumeMoun  ts :
     -  name :  my - secret - vol      #  mount   file   containing
secret   at  / secret / mountpath
        mountPath : "/ secret / mountpath "
```

Scripts and results

Generally, a template is required to run a script specified in the workflow specification. This following example shows how to do that:

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  scripts - bash -
spec :
  entrypoint :  bash - script - example
  templates :
 -  name :  bash - script - example
    steps :
   - -  name :  generate
        template :  gen - random - int - bash
   - -  name :  print
        template :  print - message
        arguments :
          parameters :
         -  name :  message
            value : "{{ steps . generate . outputs . result }}"  #
The   result   of   the   here - script

 -  name :  gen - random - int - bash
    script :
      image :  debian : 9 . 4
      command : [ bash ]
      source : |                                         #  Contents
  of   the   here - script
        cat  / dev / urandom  |  od  - N2  - An  - i  |  awk  - v   f
= 1  - v   r = 100  '{ printf  "% i \ n ",  f  +  r  * $ 1  /  65536
}'
```

```
  -  name :  gen - random - int - python
     script :
       image :  python : alpine3 . 6
       command : [ python ]
       source : |
         import   random
         i  =  random . randint ( 1 ,  100 )
         print ( i )

  -  name :  gen - random - int - javascript
     script :
       image :  node : 9 . 1 - alpine
       command : [ node ]
       source : |
         var   rand  =  Math . floor ( Math . random () *  100 );
         console . log ( rand );

  -  name :  print - message
     inputs :
       parameters :
      -  name :  message
     container :
       image :  alpine : latest
       command : [ sh , - c ]
       args : [" echo   result   was : {{ inputs . parameters .
message }}"]
```

The script keyword allows you to specify the script body by using the source tag. This action first creates a temporary file that contains the script body, and then it passes the name of the temporary file as the final parameter to the command. The command must be the interpreter that runs the script body.

The script feature can also be used to assign the standard output of running the script to a special output parameter named result. All of this allows you to use the result of running the script in the rest of the workflow specification. In the preceding example , the result is echoed by the print-message template.

Output parameters

This sample workflow templates provides a general means by which you can use the result of a step as a parameter, rather than as an artifact. This allows you to use the results from any type of step (rather than scripts) for condition tests, loops, and arguments. Output parameters work similarly to script results except that the value of the output parameters is set to the content of a generated file rather than the content of stdout.

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  output - parameter -
spec :
  entrypoint :  output - parameter
  templates :
```

```
  -  name :  output - parameter
     steps :
    - -  name :  generate - parameter
         template :  whalesay
    - -  name :  consume - parameter
         template :  print - message
         arguments :
           parameters :
          #  Pass   the   hello - param   output   from   the
generate - parameter   step   as   the   message   input   to   print
- message
          -  name :  message
             value : "{{ steps . generate - parameter . outputs .
parameters . hello - param }}"

  -  name :  whalesay
     container :
       image :  docker / whalesay : latest
       command : [ sh , - c ]
       args : [" echo  - n   hello   world  > / tmp / hello_worl  d .
txt "]  #  generate   the   content   of   hello_worl  d . txt
     outputs :
       parameters :
      -  name :  hello - param    #  name   of   output   parameter
         valueFrom :
           path : / tmp / hello_worl  d . txt   #  set   the   value
of   hello - param   to   the   contents   of   this   hello - world
. txt

  -  name :  print - message
     inputs :
       parameters :
      -  name :  message
     container :
       image :  docker / whalesay : latest
       command : [ cowsay ]
       args : ["{{ inputs . parameters . message }}"]
```

However, DAG templates use a task prefix to refer to another task. For example, `{{ tasks . generate - parameter . outputs . parameters . hello - param }}`.

Loops

- The following template iterates over a set of inputs:

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  loops -
spec :
  entrypoint :  loop - example
  templates :
 -  name :  loop - example
    steps :
   - -  name :  print - message
        template :  whalesay
        arguments :
          parameters :
         -  name :  message
            value : "{{ item }}"
```

```
        withItems :              #  invoke   whalesay   once   for
  each   item   in   parallel
       -  hello   world            #  item   1
       -  goodbye   world          #  item   2

 -  name :  whalesay
    inputs :
      parameters :
     -  name :  message
    container :
      image :  docker / whalesay : latest
      command : [ cowsay ]
      args : ["{{ inputs . parameters . message }}"]
```

- **The following template iterates over sets of items:**

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  loops - maps -
spec :
  entrypoint :  loop - map - example
  templates :
 -  name :  loop - map - example
    steps :
   - -  name :  test - linux
        template :  cat - os - release
        arguments :
          parameters :
         -  name :  image
            value : "{{ item . image }}"
         -  name :  tag
            value : "{{ item . tag }}"
        withItems :
       - {  image : ' debian ',  tag : ' 9 . 1 ' }       # item
set   1
       - {  image : ' debian ',  tag : ' 8 . 9 ' }       # item
set   2
       - {  image : ' alpine ',  tag : ' 3 . 6 ' }       # item
set   3
       - {  image : ' ubuntu ',  tag : ' 17 . 10 ' }     # item
set   4

 -  name :  cat - os - release
    inputs :
      parameters :
     -  name :  image
     -  name :  tag
    container :
      image : "{{ inputs . parameters . image }}:{{ inputs .
parameters . tag }}"
      command : [ cat ]
      args : [/ etc / os - release ]
```

- **The following template passes lists of items as parameters:**

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  loops - param - arg -
spec :
  entrypoint :  loop - param - arg - example
  arguments :
```

```
    parameters :
   -  name :  os - list                                      #  a
list   of   items
      value : |
       [
         { " image ": " debian ", " tag ": " 9 . 1 " },
         { " image ": " debian ", " tag ": " 8 . 9 " },
         { " image ": " alpine ", " tag ": " 3 . 6 " },
         { " image ": " ubuntu ", " tag ": " 17 . 10 " }
       ]

  templates :
 -  name :  loop - param - arg - example
    inputs :
      parameters :
     -  name :  os - list
    steps :
   - -  name :  test - linux
        template :  cat - os - release
        arguments :
          parameters :
         -  name :  image
            value : "{{ item . image }}"
         -  name :  tag
            value : "{{ item . tag }}"
        withParam : "{{ inputs . parameters . os - list }}"      #
parameter   specifies   the   list   to   iterate   over

 #  This   template   is   the   same   as   in   the   previous
example
 -  name :  cat - os - release
    inputs :
      parameters :
     -  name :  image
     -  name :  tag
    container :
      image : "{{ inputs . parameters . image }}:{{ inputs .
parameters . tag }}"
      command : [ cat ]
      args : [/ etc / os - release ]
```

- **The following template dynamically generates the list of items to be iterated over:**

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  loops - param - result -
spec :
  entrypoint :  loop - param - result - example
  templates :
 -  name :  loop - param - result - example
    steps :
   - -  name :  generate
        template :  gen - number - list
   #  Iterate   over   the   list   of   numbers   generated   by
the   generate   step   above
   - -  name :  sleep
        template :  sleep - n - sec
        arguments :
          parameters :
         -  name :  seconds
            value : "{{ item }}"
        withParam : "{{ steps . generate . outputs . result }}"
```

```
  # Generate a list of numbers in JSON format
  - name : gen - number - list
    script :
      image : python : alpine3 . 6
      command : [ python ]
      source : |
        import json
        import sys
        json . dump ([ i for i in range ( 20 , 31 )], sys
. stdout )

  - name : sleep - n - sec
    inputs :
      parameters :
     - name : seconds
    container :
      image : alpine : latest
      command : [ sh , - c ]
      args : [" echo sleeping for {{ inputs . parameters .
seconds }} seconds ; sleep {{ inputs . parameters . seconds }};
echo done "]
```

Conditionals

A workflow template of the Conditionals type supports conditional execution. The following is a sample template named coninflip.

```
apiVersion : argoproj . io / v1alpha1
kind : Workflow
metadata :
  generateNa me : coinflip -
spec :
  entrypoint : coinflip
  templates :
 - name : coinflip
    steps :
   # flip a coin
   - - name : flip - coin
       template : flip - coin
   # evaluate the result in parallel
   - - name : heads
       template : heads                 # call heads template
  if " heads "
       when : "{{ steps . flip - coin . outputs . result }} ==
heads "
     - name : tails
       template : tails                 # call tails template
  if " tails "
       when : "{{ steps . flip - coin . outputs . result }} ==
tails "

 # Return heads or tails based on a random number
 - name : flip - coin
    script :
      image : python : alpine3 . 6
      command : [ python ]
      source : |
        import random
        result = " heads " if random . randint ( 0 , 1 ) == 0
else " tails "
        print ( result )
```

```
  - name : heads
    container :
      image : alpine : 3 . 6
      command : [ sh , - c ]
      args : [" echo  \" it   was   heads \""]

  - name : tails
    container :
      image : alpine : 3 . 6
      command : [ sh , - c ]
      args : [" echo  \" it   was   tails \""]
```

Recursion

Workflow templates can recursively invoke each other. In the following template (a variation of the preceding coinflip template), the coin is flipped until it lands on heads.

```
apiVersion : argoproj . io / v1alpha1
kind : Workflow
metadata :
  generateNa me : coinflip - recursive -
spec :
  entrypoint : coinflip
  templates :
 - name : coinflip
    steps :
   # flip  a  coin
   - - name : flip - coin
       template : flip - coin
   # evaluate  the  result  in  parallel
   - - name : heads
       template : heads                 # call  heads  template
  if  " heads "
       when : "{{ steps . flip - coin . outputs . result }} ==
heads "
     - name : tails                     # keep  flipping  coins
  if  " tails "
       template : coinflip
       when : "{{ steps . flip - coin . outputs . result }} ==
tails "

 - name : flip - coin
    script :
      image : python : alpine3 . 6
      command : [ python ]
      source : |
        import  random
        result  = " heads "  if  random . randint ( 0 , 1 ) ==  0
else  " tails "
        print ( result )

 - name : heads
    container :
      image : alpine : 3 . 6
      command : [ sh , - c ]
```

```
        args : [" echo  \" it   was   heads \""]
```

The following shows the result of several times of running templates to flip the coin:

```
ags   get   coinflip - recursive - tzcb5

STEP                               PODNAME
MESSAGE
#  coinflip - recursive - vhph5
├---#  flip - coin                coinflip - recursive - vhph5 -
2123890397
└-·-#  heads                    coinflip - recursive - vhph5 -
128690560
  └-o  tails

STEP                                PODNAME
  MESSAGE
#  coinflip - recursive - tzcb5
├---#  flip - coin                  coinflip - recursive - tzcb5 -
322836820
└-·-o  heads
  └-#  tails
    ├---#  flip - coin              coinflip - recursive - tzcb5 -
1863890320
    └-·-o  heads
      └-#  tails
        ├---#  flip - coin          coinflip - recursive - tzcb5 -
1768147140
        └-·-o  heads
          └-#  tails
            ├---#  flip - coin      coinflip - recursive - tzcb5 -
4080411136
            └-·-#  heads          coinflip - recursive - tzcb5 -
4080323273
              └-o  tails
```

In the first round to run the template to flip the coin, the coin immediately lands on heads and and the coin is no longer flipped. In the second round to flip the coin, the coin lands on tails three times before landing on heads at which time the coin is no longer flipped.

Exit handlers

An exit handler is a template run at the end of the workflow, regardless of whether the workflow succeeded or failed.

Exit handlers can be used if you want to perform any of the following actions:

- Clean up after a workflow is run.
- Send notifications of workflow status (for example, emails or Slack messages).
- Post the pass or fail status to a WebHook result (for example, a GitHub build result).

- **Resubmit a workflow or submit a new workflow.**

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  exit - handlers -
spec :
  entrypoint :  intentiona  l - fail
  onExit :  exit - handler                   #  invoke   exit -
hander   template   at   end   of   the   workflow
  templates :
 #  primary   workflow   template
 -  name :  intentiona  l - fail
    container :
      image :  alpine : latest
      command : [ sh , - c ]
      args : [" echo   intentiona  l   failure ;  exit   1 "]

 #  Exit   handler   templates
 #  After   the   completion   of   the   entrypoint   template ,
the   status   of   the
 #  workflow   is   made   available   in   the   global   variable
{{ workflow . status }}.
 # {{ workflow . status }}  will   be   one   of :  Succeeded ,
Failed ,  Error
 -  name :  exit - handler
    steps :
   - -  name :  notify
        template :  send - email
     -  name :  celebrate
        template :  celebrate
        when : "{{ workflow . status }} ==  Succeeded "
     -  name :  cry
        template :  cry
        when : "{{ workflow . status }} !=  Succeeded "
 -  name :  send - email
    container :
      image :  alpine : latest
      command : [ sh , - c ]
      args : [" echo   send   e - mail : {{ workflow . name }} {{
workflow . status }}"]
 -  name :  celebrate
    container :
      image :  alpine : latest
      command : [ sh , - c ]
      args : [" echo   hooray !"]
 -  name :  cry
    container :
      image :  alpine : latest
      command : [ sh , - c ]
      args : [" echo   boohoo !"]
```

Timeouts

A workflow template of the timeouts type can be used to limit the timeout of a workflow. In such a template, you must set the variable `activeDead lineSecond s` for a timeout value to be specified.

```
#  To   enforce   a   timeout   for   a   container   template ,
 specify   a   value   for   activeDead  lineSecond  s .
```

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  timeouts -
spec :
  entrypoint :  sleep
  templates :
 -  name :  sleep
    container :
      image :  alpine : latest
      command : [ sh , - c ]
      args : [" echo   sleeping   for   1m ;  sleep   60 ;  echo
done "]
    activeDead  lineSecond  s :  10            #  terminate
container   template   after   10   seconds
```

Volumes

In the following example, a volume is dynamically created, and then used in a two-step workflow.

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  volumes - pvc -
spec :
  entrypoint :  volumes - pvc - example
  volumeClai  mTemplates :                 #  define   volume ,  same
  syntax   as   k8s   Pod   spec
 -  metadata :
      name :  workdir                      #  name   of   volume
claim
    spec :
      accessMode  s : [ " ReadWriteO  nce " ]
      resources :
        requests :
          storage :  1Gi                   #  Gi  =>  1024  *  1024
*  1024

  templates :
 -  name :  volumes - pvc - example
    steps :
   - -  name :  generate
        template :  whalesay
   - -  name :  print
        template :  print - message

 -  name :  whalesay
    container :
      image :  docker / whalesay : latest
      command : [ sh , - c ]
      args : [" echo   generating   message   in   volume ;  cowsay
hello   world  |  tee  / mnt / vol / hello_worl  d . txt "]
     #  Mount   workdir   volume   at  / mnt / vol   before
invoking   docker / whalesay
      volumeMoun  ts :                     #  same   syntax   as
k8s   Pod   spec
     -  name :  workdir
        mountPath : / mnt / vol

 -  name :  print - message
    container :
```

```
      image :  alpine : latest
      command : [ sh , - c ]
      args : [" echo   getting   message   from   volume ;  find  /
mnt / vol ;  cat  / mnt / vol / hello_worl  d . txt "]
     #  Mount   workdir   volume   at  / mnt / vol   before
invoking   docker / whalesay
      volumeMoun  ts :                    #  same   syntax   as
k8s   Pod   spec
     -  name :  workdir
        mountPath : / mnt / vol
```

Volumes help you move large amounts of data from one step in a workflow to another . Depending on the system, some volumes can be accessed from multiple steps at the same time.

If you want to access an existing volume, instead of dynamically create or destroy a volume, you can use or modify the following sample volume as needed:

```
#  Define   Kubernetes   PVC
 kind :  Persistent  VolumeClai  m
 apiVersion :  v1
 metadata :
   name :  my - existing - volume
 spec :
   accessMode  s : [ " ReadWriteO  nce " ]
   resources :
     requests :
       storage :  1Gi

---
 apiVersion :  argoproj . io / v1alpha1
 kind :  Workflow
 metadata :
   generateNa  me :  volumes - existing -
 spec :
   entrypoint :  volumes - existing - example
   volumes :
  #  Pass   my - existing - volume   as   an   argument   to   the
 volumes - existing - example   template
  #  Same   syntax   as   k8s   Pod   spec
  -  name :  workdir
     persistent  VolumeClai  m :
       claimName :  my - existing - volume

   templates :
  -  name :  volumes - existing - example
     steps :
    - -  name :  generate
         template :  whalesay
    - -  name :  print
         template :  print - message

  -  name :  whalesay
     container :
       image :  docker / whalesay : latest
       command : [ sh , - c ]
       args : [" echo   generating   message   in   volume ;  cowsay
 hello   world  |  tee  / mnt / vol / hello_worl  d . txt "]
       volumeMoun  ts :
      -  name :  workdir
```

```
        mountPath : / mnt / vol

  -  name :  print - message
     container :
       image :  alpine : latest
       command : [ sh , - c ]
       args : [" echo   getting   message   from   volume ;  find  /
mnt / vol ;  cat  / mnt / vol / hello_worl  d . txt "]
       volumeMoun  ts :
      -  name :  workdir
        mountPath : / mnt / vol
```

Daemon containers

Workflows can start containers (also known as daemon containers) that run in the backend while the workflow continues to be run. The daemons are automatica lly destroyed when the workflow exits the template scope in which the daemon is invoked. Daemon containers can be used to start services to be tested, or can be directly used in a fixture test or other tests.

Daemon containers can also be used to run large simulations to set a database as a daemon for collecting and organizing the results. The advantage of daemons over sidecars is that daemons can be run over multiple steps or even the entire workflow.

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  daemon - step -
spec :
  entrypoint :  daemon - example
  templates :
  -  name :  daemon - example
     steps :
    - -  name :  influx
        template :  influxdb               #  start   an   influxdb
as   a   daemon  ( see   the   influxdb   template   spec   below )

    - -  name :  init - database              #  initialize
influxdb
        template :  influxdb - client
        arguments :
          parameters :
         -  name :  cmd
            value :  curl  - XPOST  ' http ://{{ steps . influx .
ip }}: 8086 / query ' -- data - urlencode  " q = CREATE   DATABASE
mydb "

    - -  name :  producer - 1                 #  add   entries   to
influxdb
        template :  influxdb - client
        arguments :
          parameters :
         -  name :  cmd
            value :  for   i   in  $( seq   1   20 );  do   curl  -
XPOST  ' http ://{{ steps . influx . ip }}: 8086 / write ? db = mydb
' - d  " cpu , host = server01 , region = uswest   load =$ i " ;
sleep  . 5  ;  done
```

```
    - name : producer - 2                # add entries to
influxdb
        template : influxdb - client
        arguments :
          parameters :
         - name : cmd
            value : for i in $( seq 1 20 ); do curl -
XPOST ' http ://{{ steps . influx . ip }}: 8086 / write ? db = mydb
' - d " cpu , host = server02 , region = uswest load =$(( RANDOM
% 100 ))" ; sleep . 5 ; done
    - name : producer - 3                # add entries to
influxdb
        template : influxdb - client
        arguments :
          parameters :
         - name : cmd
            value : curl - XPOST ' http ://{{ steps . influx . ip
}}: 8086 / write ? db = mydb ' - d ' cpu , host = server03 , region
= useast load = 15 . 4 '

   - - name : consumer                 # consume intries
from influxdb
        template : influxdb - client
        arguments :
          parameters :
         - name : cmd
            value : curl -- silent - G http ://{{ steps . influx
. ip }}: 8086 / query ? pretty = true -- data - urlencode " db =
mydb " -- data - urlencode " q = SELECT * FROM cpu "

 - name : influxdb
    daemon : true                        # start influxdb as
a daemon
    container :
      image : influxdb : 1 . 2
      restartPol icy : Always              # restart container
if it fails
      readinessP robe :                  # wait for
readinessP robe to succeed
        httpGet :
          path : / ping
          port : 8086

 - name : influxdb - client
    inputs :
      parameters :
     - name : cmd
    container :
      image : appropriat e / curl : latest
      command : ["/ bin / sh ", "- c "]
      args : ["{{ inputs . parameters . cmd }}"]
      resources :
        requests :
          memory : 32Mi
          cpu : 100m
```

DAG templates use the tasks prefix to refer to another task. For example, `{{ tasks . influx . ip }}`.

Sidecars

A sidecar is a container that is run in the same pod where the main container is executed. Sidecars help you create a pod that contains multiple containers.

The following workflow template of the sidecar type creates a sidecar container that runs Nginx as a simple web server. Containers come up in random order. Therefore, the main container polls the Nginx container until it is ready to response requests. We recommend that you use this design pattern for multi-container systems. That is, before you run the main code, you must wait for all of the required services to come up.

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  sidecar - nginx -
spec :
  entrypoint :  sidecar - nginx - example
  templates :
 -  name :  sidecar - nginx - example
    container :
      image :  appropriat  e / curl
      command : [ sh , - c ]
     #  Try   to   read   from   nginx   web   server   until   it
comes   up
      args : [" until  ` curl  - G  ' http :// 127 . 0 . 0 . 1 /'
>& / tmp / out `;  do   echo   sleep  &&  sleep   1 ;  done  &&  cat
/ tmp / out "]
   #  Create   a   simple   nginx   web   server
    sidecars :
   -  name :  nginx
      image :  nginx : 1 . 13
```

Kubernetes resources

If you want to manage Kubernetes resources by using workflows, you can use a resource template, which allows you to create, delete, or updated any type of Kubernetes resources.

```
#  in   a   workflow .  The   resource   template   type   accepts
 any   k8s   manifest
# ( including   CRDs )  and   can   perform   any   kubectl   action
 against   it  ( e . g .  create ,
#  apply ,  delete ,  patch ).
 apiVersion :  argoproj . io / v1alpha1
 kind :  Workflow
 metadata :
   generateNa  me :  k8s - jobs -
 spec :
   entrypoint :  pi - tmpl
   templates :
  -  name :  pi - tmpl
     resource :                   #  indicates   that   this   is   a
   resource   template
```

```
      action :  create             #  can   be   any   kubectl
action  ( e . g .  create ,  delete ,  apply ,  patch )
     #  The   successCon  dition   and   failureCon  dition   are
optional   expression  s .
     #  If   failureCon  dition   is   true ,  the   step   is
considered   failed .
     #  If   successCon  dition   is   true ,  the   step   is
considered   successful .
     #  They   use   kubernetes   label   selection   syntax   and
can   be   applied   against   any   field
     #  of   the   resource  ( not   just   labels ).  Multiple   AND
  conditions   can   be   represente  d   by   comma
     #  delimited   expression  s .
     #  For   more   details :  https :// kubernetes . io / docs /
concepts / overview / working - with - objects / labels /
      successCon  dition :  status . succeeded  >  0
      failureCon  dition :  status . failed  >  3
      manifest : |              # put   your   kubernetes   spec
here
        apiVersion :  batch / v1
        kind :  Job
        metadata :
          generateNa  me :  pi - job -
        spec :
          template :
            metadata :
              name :  pi
            spec :
              containers :
             -  name :  pi
                image :  perl
                command : [" perl ",  "- Mbignum = bpi ", "- wle ", "
print   bpi ( 2000 )"]
              restartPol  icy :  Never
          backoffLim  it :  4
```

Resources created with this method are independent of the workflow. If you want the resource to be deleted when you delete the workflow, you can use Kubernetes garbage collection and the workflow resources as an owner reference.

Note:

- When you patch the Kubernetes resources, the resources gain the `mergeStrat egy` attribute, which can be `strategy` , `merge` , or `json` . By default, `strategy` is used.
- `strategy` cannot be used to patch custom resources. You must use one of the other two `mergeStrat egy` values. For example, you have defined a CronTab of Custom Resource Definition as follows:

```
apiVersion : " stable . example . com / v1 "
kind :  CronTab
spec :
  cronSpec : "* * * * */ 5 "
```

```
    image :  my - awesome - cron - image
```

You can modify the preceding CronTab by using the following workflow:

```
apiVersion :  argoproj . io / v1alpha1
kind :  Workflow
metadata :
  generateNa  me :  k8s - patch -
spec :
  entrypoint :  cront - tmpl
  templates :
 -  name :  cront - tmpl
    resource :
      action :  patch
      mergeStrat  egy :  merge                  #  Must   be   one
  of  [ strategic   merge   json ]
      manifest : |
        apiVersion : " stable . example . com / v1 "
        kind :  CronTab
        spec :
          cronSpec : "* * * * */ 10 "
          image :  my - awesome - cron - image
```

Reference

- For information about more resources, see Argo workflow templates by example.
- For information about all the sample templates, see Sample templates.

5.3 Enable the workflow UI

This topic describes how to enable and access the workflow UI by creating an Ingress. By using the workflow UI, you can view the status of all workflows, and the container logs of each step of a workflow.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- The Master node of the Kubernetes cluster is accessible. For more information, see Connect to a Kubernetes cluster by using kubectl.

Procedure

1. Run an htpasswd command to generate the file *auth* .

   Note:
   In this file, you can set the password used to access the workflow UI.

   ```
   $  htpasswd  - c  auth  workflow
    New  password : < workflow >
   ```

```
New   password :
Re - type   new   password :
Adding   password   for   user   workflow
```

2. Run the following command to create a Secret that is used to store the `auth` file in the target Kubernetes cluster.

```
$ kubectl   create   secret   generic   workflow - basic - auth   --
from - file = auth  - n   argo
```

3. Create the file `ingress yaml`, and then copy the following code to the file:

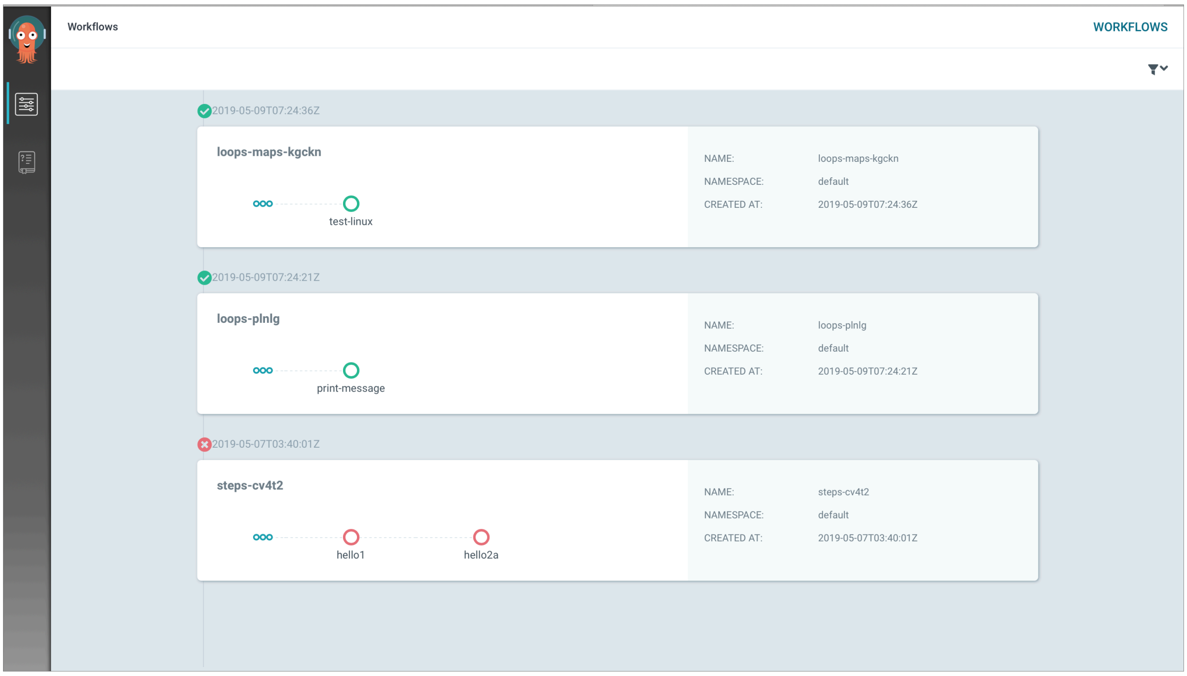
```
apiVersion :  extensions / v1beta1
kind :  Ingress
metadata :
  name :  workflow - ingress
  namespace :  argo
  annotation  s :
   #  type   of   authentica  tion
    nginx . ingress . kubernetes . io / auth - type :  basic
   #  name   of   the   secret   that   contains   the   user /
password   definition  s
    nginx . ingress . kubernetes . io / auth - secret :  workflow -
basic - auth
   #  message   to   display   with   an   appropriat  e   context
  why   the   authentica  tion   is   required
    nginx . ingress . kubernetes . io / auth - realm : '
Authentica  tion   Required  -  workflow '
spec :
  rules :
 -  host :  workflow .< yourTestHo  st >
    http :
      paths :
     -  path : /
        backend :
          serviceNam  e :  argo - ui
          servicePor  t :  80
```

Note:

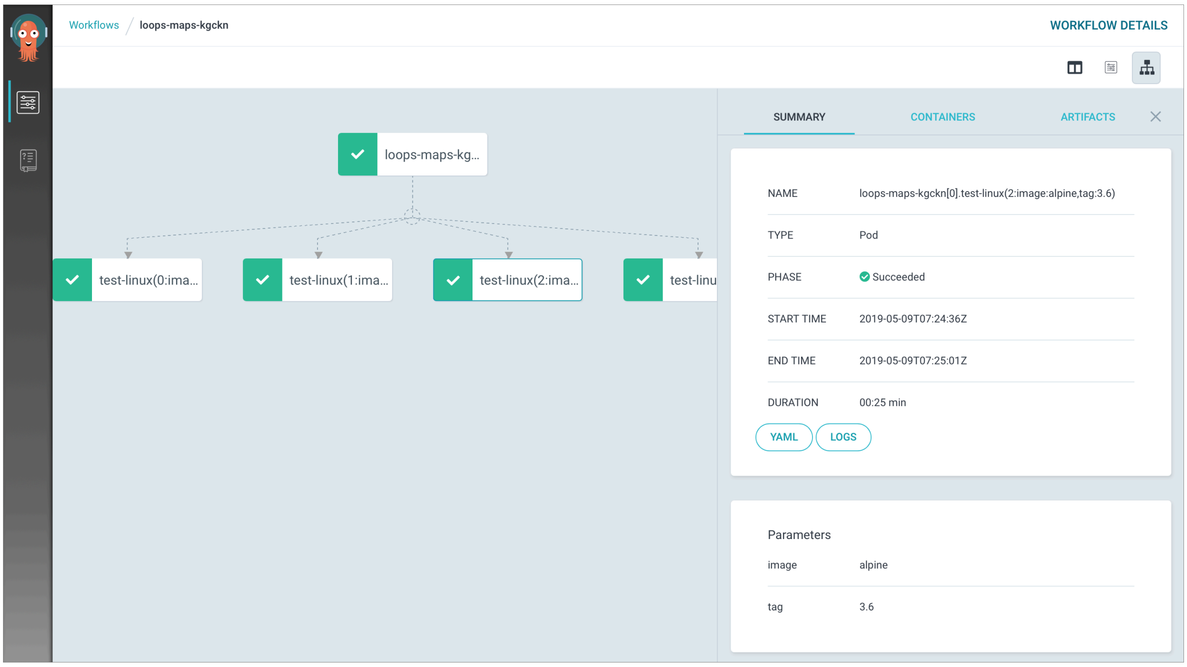
You must replace `< yourTestHo st >` with your cluster address (That is, the value of `Testing Domain` in the Cluster Information area. For example, cfb131.cn-zhangjiakou.alicontainer.com).

4. Run the `kubectl apply - f ingress . yaml` command to create an Ingress named workflow-ingress.

5. In your browser, enter `workflow.<yourTestHost>`, and then enter the password to view the page of workflow UI.



6. View the status of the target workflow.



5.4 Introduction to AGS CLI

This topic describes the features of Alibaba Cloud Genomics Compute Service (AGS) and how to download and configure AGS. By default, AGS calls Alibaba Cloud services by using Alibaba Cloud access keys. AGS works with Log Service to collect pod logs. To use this function, you must select to enable Log Service when you create a cluster.

Features

- Fully compatible to argo commands
- Works with Log Service, allowing you to view logs after a pod is deleted
- Compatible to kubectl commands, allowing you to manage clusters by using kubectl
- Provides `install` and `uninstall` commands, allowing you to install or uninstall resources as needed

  > **Note:**
  >
  > - You can run the `ags install` command to install resources.
  > - You can run the `ags uninstall` command to uninstall resources.
- Provides the `get workflow` command, allowing you to view the resource usage
- Provides YAML templates that allow special characters such as underscores (_)
- Provides security contexts
- Synchronizes pod status (such as pending and failed) with workflows
- Uses YAML templates to customize retries
- Retries an entire workflow when the workflow fails at any point
- Supports ECI Serverless Kubernetes architecture

Download and installation

To install AGS and configure relevant permissions, run the following command:

```
wget  http :// ags - hub . oss - cn - hangzhou . aliyuncs . com /
ags - linux  &&  chmod  + x  ags - linux  &&  mv  ags - linux  /
usr / local / bin / ags
```

> **Note:**
>
> - You can run the `ags config init` command to enter required information in the CLI. After the system is initialized, the configuration files are saved to the *~/. ags / config* file. You can run the `ags config show` command to display the configured information. In the configured information, the `AccessKeyS ecret` is encrypted.

- To collect logs by using Log Service, you must first configure `ags config` . We recommend that you create an access key for the CLI and grant relevant permissions for Log Service.

If you are using a managed Kubernetes cluster, you can connect to the Kubernetes cluster by using kubectl and run the following command to use AGS through Cloud Shell:

```
wgethttp :// ags - hub . oss - cn - hangzhou . aliyuncs . com / ags
- linux && chmod + x ags - linux && mv ags - linux / usr /
local / bin / ags
```

Available commands in AGS CLI

The available commands in AGS CLI are as follows:

```
[ root @ iZwz92q9h3 6kv8posr0i 6uZ ~]# ags
 ags is the command line interface to Alibaba Cloud
 Genomics Compute Service

 Usage :
  ags [ flags ]
  ags [ command ]

 Available Commands :
  completion    output shell completion code for the
 specified shell ( bash or zsh )
  config        setup ags client necessary info
  delete        delete a workflow and its associated pods
  get           display details about a workflow
  help          Help about any command
  install       install ags
  kubectl       kubectl command
  lint          validate a file or directory of workflow
 manifests
  list          list workflows
  logs          view logs of a workflow
  resubmit      resubmit a workflow
  resume        resume a workflow
  retry         retry a workflow
  submit        submit a workflow
  suspend       suspend a workflow
  terminate     terminate a workflow
  uninstall     uninstall ags
  version       Print version informatio n
  wait          waits for a workflow to complete
  watch         watch a workflow until it completes

 Flags :
     -- as string                     Username to impersonat
 e for the operation
     -- as - group stringArra y             Group to
 impersonat e for the operation , this flag can be
 repeated to specify multiple groups .
     -- certificat e - authority string     Path to a cert
 file for the certificat e authority
     -- client - certificat e string        Path to a client
   certificat e file for TLS
```

```
    -- client - key   string                Path   to   a   client
key   file   for   TLS
    -- cluster   string                   The   name   of   the
kubeconfig   cluster   to   use
    -- context   string                   The   name   of   the
kubeconfig   context   to   use
- h , -- help                             help   for   ags
    -- insecure - skip - tls - verify         If   true ,  the
  server ' s   certificat  e   will   not   be   checked   for
validity .  This   will   make   your   HTTPS   connection  s
insecure
    -- kubeconfig   string                Path   to   a   kube
config .  Only   required   if   out - of - cluster
- n , -- namespace   string                If   present ,  the
namespace   scope   for   this   CLI   request
    -- password   string                  Password   for   basic
authentica  tion   to   the   API   server
    -- request - timeout   string           The   length   of
time   to   wait   before   giving   up   on   a   single   server
request .  Non - zero   values   should   contain   a   correspond
ing   time   unit   ( e . g .  1s ,  2m ,  3h ).  A   value   of
zero   means   don ' t   timeout   requests . ( default  " 0 ")
    -- server   string                    The   address   and   port
  of   the   Kubernetes   API   server
    -- token   string                     Bearer   token   for
authentica  tion   to   the   API   server
    -- user   string                      The   name   of   the
kubeconfig   user   to   use
    -- username   string                  Username   for   basic
authentica  tion   to   the   API   server

Use  " ags  [ command ] -- help "  for   more   informatio  n   about
  a   command .
```

6 Network management

6.1 ACK network overview

This topic describes networks supported by Alibaba Cloud Container Service for Kubernetes (ACK). Specifically, the interconnections supported within container networks, functions of container networks, infrastructures of container networks, the network plugin Terway, and network policies.

Interconnections within container networks

ACK provides stable and high-performance container networks by integrating Kubernetes networks and Alibaba Cloud Virtual Private Cloud (VPC) networks. The following features are supported within the interconnections of a container network:

- Pods that can access each other in a Kubernetes cluster.
- A pod that can access a service in a Kubernetes cluster.
- An Elastic Compute Service (ECS) instance that can access a service.
- A pod and an ECS instance that are mutually accessible in the same VPC, if a valid security group rule is set.

Functions of container networks

Service

In Kubernetes, a service is an object, which is assigned with a fixed IP address. The service forwards its received traffic to the corresponding pods according to label selectors. In addition, services in Kubernetes can work as load balancers for pods.

Services in Kubernetes solve the troubles caused by the following issues:

- Pods may be inaccessible because controllers in Kubernetes delete pods and then recreate new pods at any time.
- The IP address of a pod cannot be obtained. A pod is assigned with an IP address only after the pod is started.
- It is not feasible to access pods one by one because an application consists of a group of pods that run the same image.

For more information, see Create a service.

Ingress

IP addresses of services and pods can only be accessed within a Kubernetes cluster. Any request outside a Kubernetes cluster must be forwarded by a load balancer to the NodePort that is exposed by the destination service on a node, and then kube-proxy forwards the request to the corresponding pod by using an edge router or deserts the request. An Ingress in Kubernetes is an object that provides routing rules for the requests that enter into a Kubernetes cluster.

An Ingress can be configured to provide services with externally-reachable URLs , load balance traffic, terminate SSL / TLS, HTTP routes, and more. To implement these Ingress rules, the cluster administrator must deploy an Ingress controller. The Ingress controller listens to the changes occurred to the Ingress and services , configures load balancer according to the Ingress rules, and provides access endpoints.

The Ingress function is implemented by the following parts:

- Nginx: works as a load balancer to distribute requests to pods.
- Ingress controller: obtains the IP address of the pod that corresponds to the service by accessing the cluster API server, and then adds the IP address to the configuration file of Nginx.
- Ingress: creates a virtual machine for Nginx.

For more information, see Create an Ingress in the Container Service console.

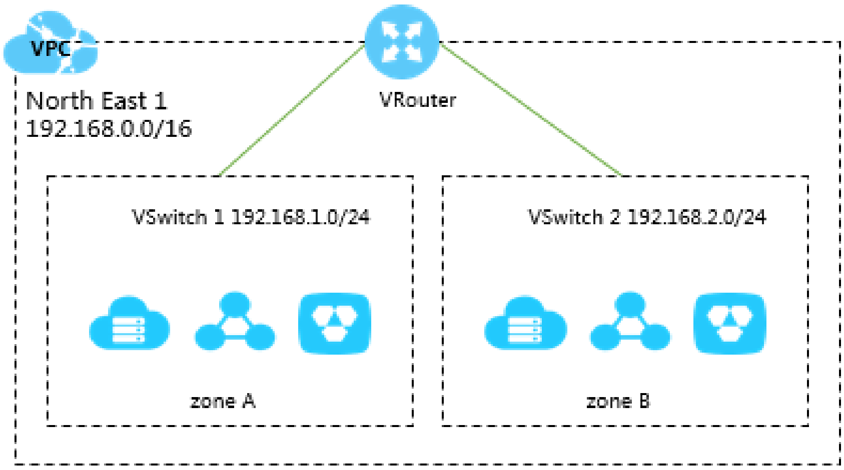
Infrastructures of container networks

Infrastructures of container networks consist of VPCs and SLB instances.

VPC

Virtual Private Cloud (VPC) is a type of private network developed by Alibaba Cloud. Different VPCs are logically isolated from other virtual networks in Alibaba Cloud. In a VPC, you can create and manage instances of cloud resources, such as ECS instances (cloud servers), RDS instances (cloud databases), and SLB instances.

A VPC consists of a private CIDR block, a VRouter, and at least a VSwitch.



SLB

By setting a virtual service address, SLB virtualizes added ECS instances into an application service pool that has high performance and high availability, and distributes client requests to ECS instances in the server pool based on forwarding rules.

SLB also checks the health status of added backend servers, and automatically isolates abnormal ECS instances to eliminate Single Point of Failures (SPOFs), improving the overall service capability of your application. Additionally, working with Alibaba Anti-DDoS, SLB can defend DDoS attacks.

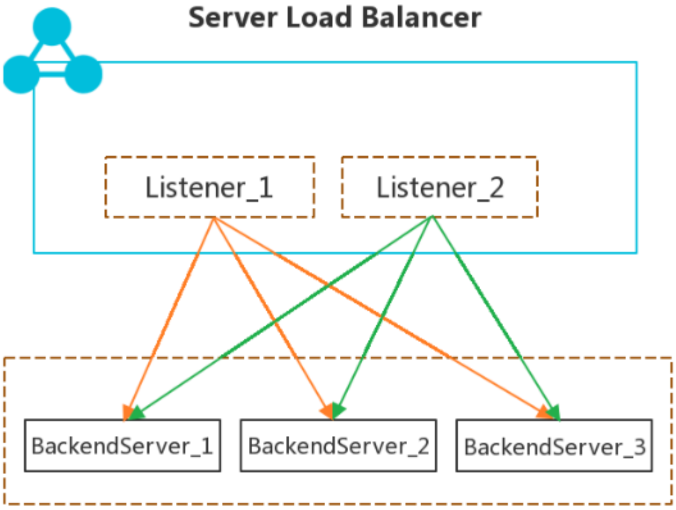
SLB consists of the following components:

- SLB instances

  An SLB instance is a running load balancing service that distributes incoming traffic to backend servers. To use the SLB service, you must create an SLB instance, and then configure the instance with at least one listener and two backend servers.
- Listeners

  A listener checks client requests and forwards the requests to backend servers according to the configured rules. It also performs health checks on backend servers.
- Backend servers

  Backend servers are the ECS instances added to an SLB instance to process the distributed requests. You can add ECS instances to the default server group, a VServer group, or an active/standby server group for better management.



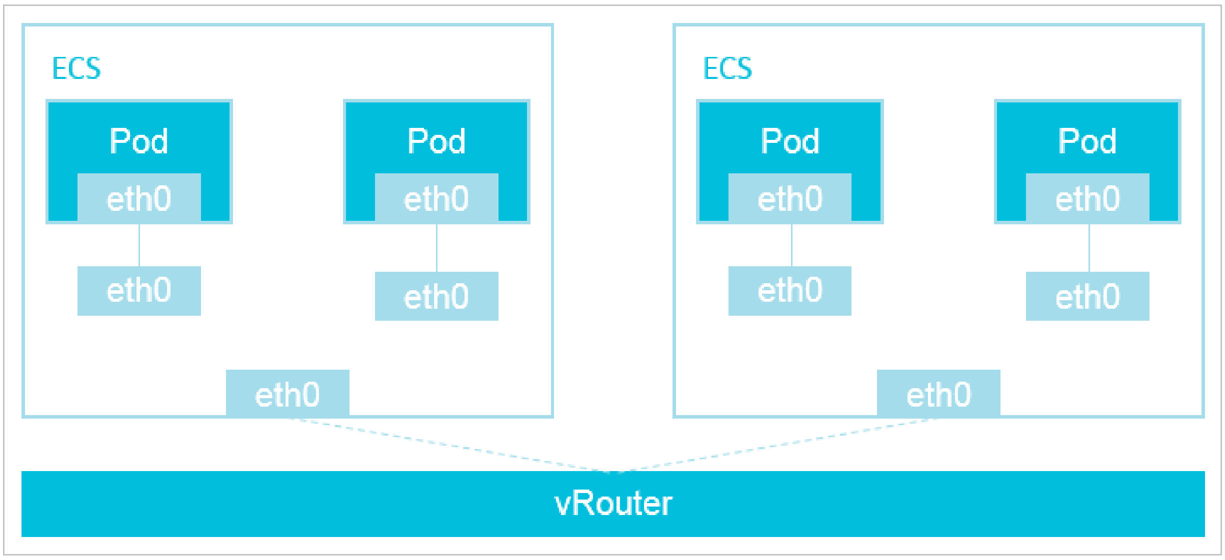
Network plugin Terway

Terway is a network plugin developed by ACK. It is fully compatible with the Flannel plugin.

The following are features of Terway:

- Terway allocates Alibaba Cloud Elastic Network Interfaces (ENIs) to containers.
- Terway defines the access policies between containers according to the Kubernetes network policies. It is compatible with the Calico network policies.

The container network supported by Terway shows higher communication performance because no VXLAN or any other tunnel technology is used to encapsulat e packets. Specifically, in the container network that the Terway network plugin is installed, each pod in a Kubernetes cluster has a network stack and IP address. When pods that run on one ECS instance communicate with each other, packets are forwarded within the ECS instance. When pods that run on different ECS instance communicate with each other, packets are forwarded by a VRouter of the VPC where the Kubernetes cluster is located.



Network policy

A network policy is a specification of how pods are allowed to communicate with each other and other network endpoints.

In Kubernetes, the object used to configure a network policy is `NetworkPol icy`. It uses labels to select pods, and defines rules which specify what traffic is allowed to the selected pods. For more information, see Use a network policy.

6.2 Access services by using Server Load Balancer

This topic describes how to access services by using Alibaba Cloud Server Load Balancer (SLB).

Check the version of the Cloud Controller Manager

By default, if you specify an existing SLB instance for a Kubernetes cluster that uses the Cloud Controller Manager of V1.9.3 or later, the system does not configure listeners for this SLB instance. You must set the parameter `service . beta . kubernetes . io / alicloud - loadbalanc er - force - override - listeners : " true "` to enable the system to listen to the SLB instance, or manually configure listeners for this SLB instance.

To view the version of the Cloud Controller Manager, run the following command:

```
root @ master  #  kubectl  get  po  - n  kube - system  - o  yaml
| grep  image :| grep  cloud - con | uniq
```

```
  image :  registry - vpc . cn - hangzhou . aliyuncs . com / acs /
cloud - controller - manager - amd64 : v1 . 9 . 3
```

Use a command-line tool

Method 1

1. **Create an Nginx application by using a command-line tool.**

```
root @ master  #  kubectl   run   nginx  -- image = registry .
aliyuncs . com / acs / netdia : latest
root @ master  #  kubectl   get   po
NAME                                     READY      STATUS
RESTARTS    AGE
nginx - 2721357637 - dvwq3                   1 / 1        Running
   1            6s
```

2. **Create an SLB service for the Nginx application and specify `type = LoadBalancer` to expose the Nginx service to the Internet.**

```
root @ master  #  kubectl   expose   deployment   nginx  -- port =
80  -- target - port = 80  -- type = LoadBalanc  er
root @ master  #  kubectl   get   svc
NAME                  CLUSTER - IP        EXTERNAL - IP
PORT ( S )                        AGE
nginx                 172 . 19 . XX . XX     101 . 37 . XX . XX
   80 : 31891 / TCP                     4s
```

3. **Visit `http :// 101 . 37 . XX . XX` in a browser to access your Nginx service.**

Method 2

1. **Save the following yml code to the `nginx - svc . yml` file:**

```
apiVersion :  v1
kind :  Service
metadata :
  labels :
    run :  nignx
  name :  nginx - 01
  namespace :  default
spec :
  ports :
 -  port :  80
    protocol :  TCP
    targetPort :  80
  selector :
    run :  nginx
  type :  LoadBalanc  er
```

2. **Run the `kubectl   apply  - f   nginx - svc . yml` command.**

```
root @ master  #  kubectl   apply  - f   nginx - svc . yml
root @ master  #  kubectl   get   service
NAME             TYPE              CLUSTER - IP        EXTERNAL - IP
       PORT ( S )         AGE9d
```

```
ngi - 01nx          LoadBalanc  er     172 . 19 . XX . XX      101 .
37 . XX . XX        80 : 32325 / TCP     3h
```

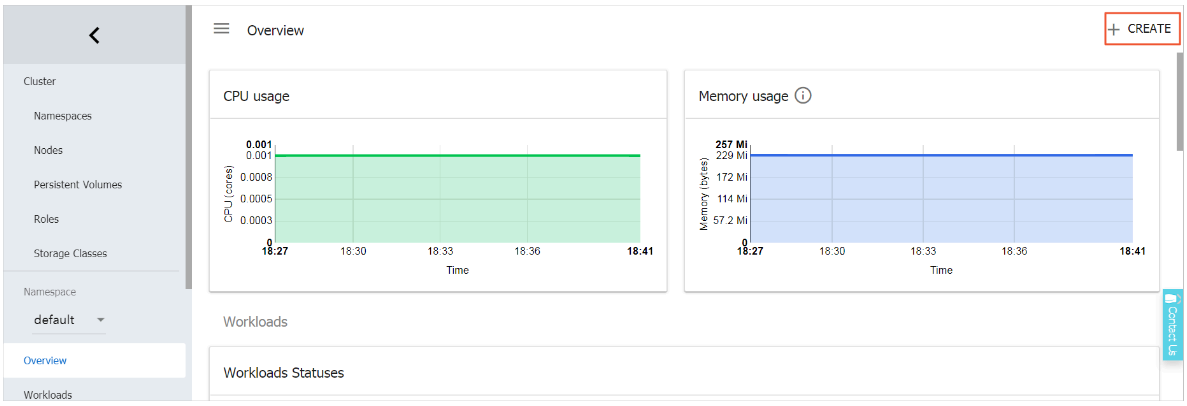
3. Visit `http :// 101 . 37 . XX . XX` in a browser to access your Nginx service.

Use the Kubernetes dashboard

1. Save the following yml code to the `nginx - svc . yml` file:

```
apiVersion :  v1
kind :  Service
metadata :
  labels :
    run :  nginx
  name :  http - svc
  namespace :  default
spec :
  ports :
  - port :  80
    protocol :  TCP
    targetPort :  80
  selector :
    run :  nginx
  type :  LoadBalanc  er
```

2. Log on to the Container Service console and click Dashboard on the right of the target cluster.
3. Click CREATE in the upper-right corner to create an application.

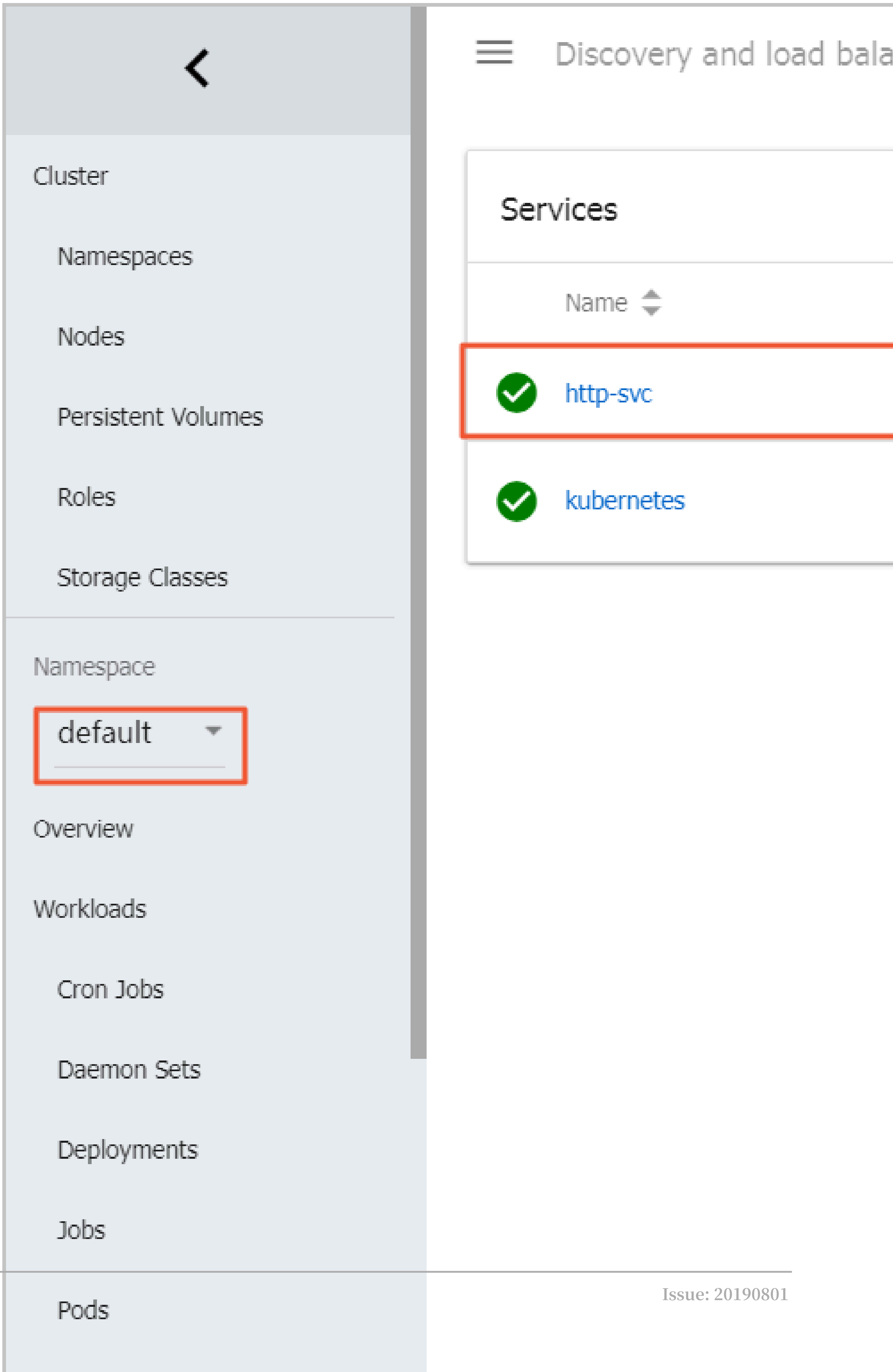


4. Click the CREATE FROM FILE tab. Select the *nginx - svc . yml* file you saved.
5. Click UPLOAD.

   An SLB instance that points to the created Nginx application is created. The service name is `http - svc`.

6. In the left-side navigation pane on the dashboard page, select the default namespace, and then click `Services`.

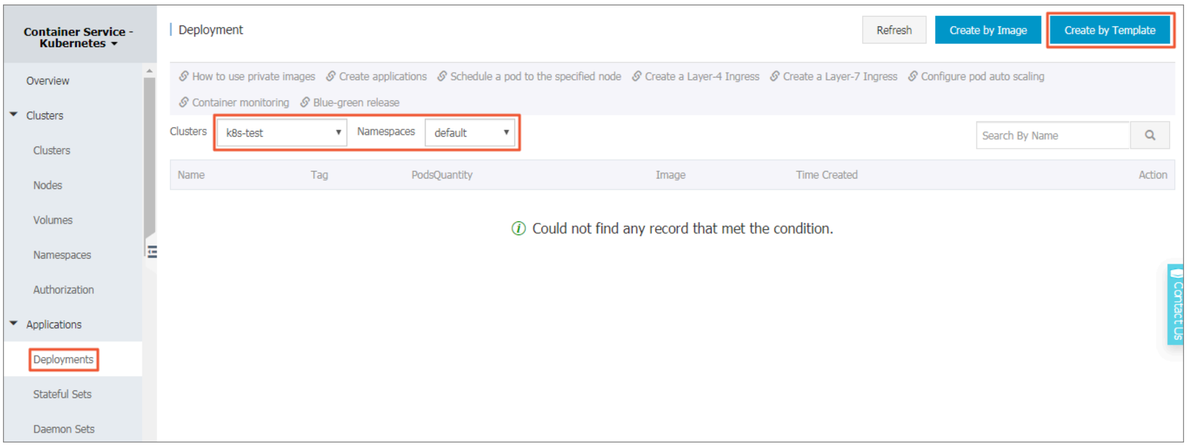
   You can view the created Nginx service `http - svc` and its IP address.


Discovery and load bala
Cluster
Namespaces
Nodes
Persistent Volumes
Roles
Storage Classes
Namespace
default
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Services
Name
http-svc
kubernetes

7. Open this address in your browser to access the service.

Use the Container Service console

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Application > Deployment.
3. Select the target cluster and namespace, and then click Create by Template in the upper-right corner.

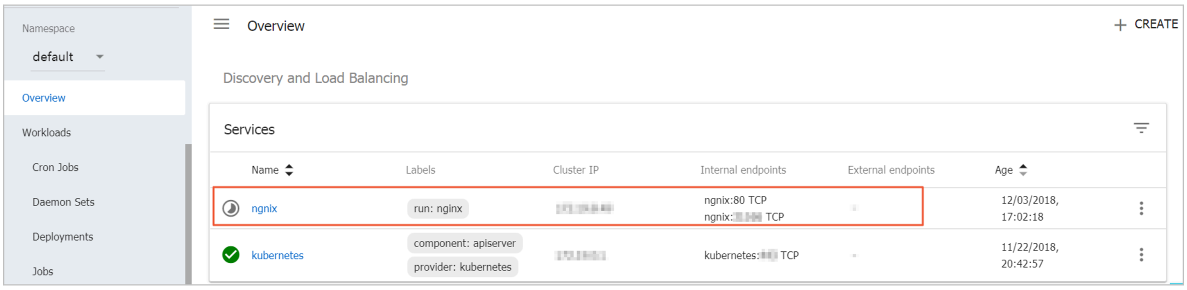


4. Select the custom Resource Type and then copy the following code to the Template.

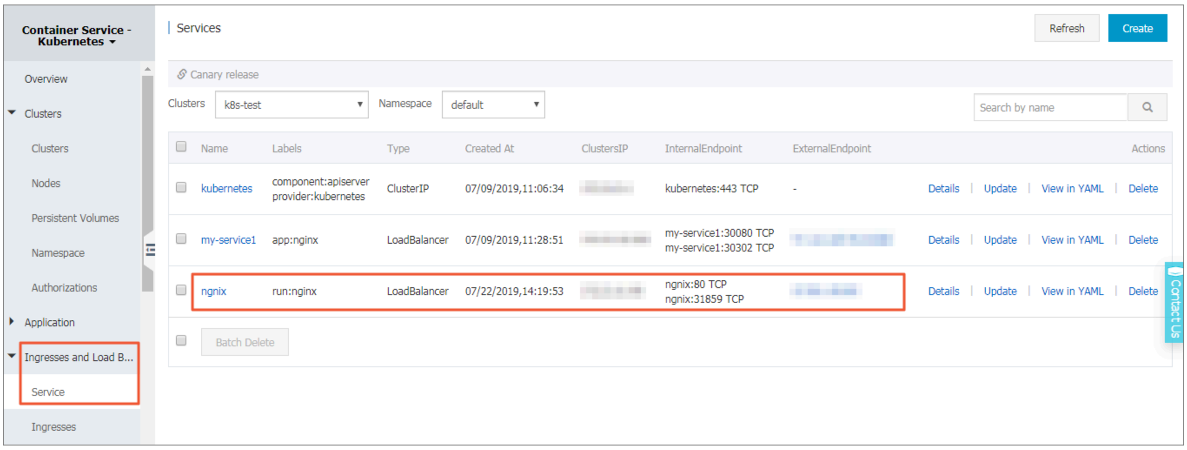
```
apiVersion :  v1
kind :  Service
metadata :
  labels :
    run :  nginx
  name :  ngnix
  namespace :  default
spec :
  ports :
 - port :  80
    protocol :  TCP
    targetPort :  80
  selector :
    run :  nginx
  type :  LoadBalanc   er
```

5. Click DEPLOY.

6. Click Kubernetes Dashboard to check the deployment progress on the dashboard page.



Alternatively, choose Ingresses and Load Balancing > Service in the left-side navigation pane, and select the target cluster and namespace to view the deployed service.



More information

Alibaba Cloud SLB also supports a lot of parameters such as health checks, billing methods, and SLB types. For more information, see SLB configuration parameters.

Annotations

Alibaba Cloud supports plenty of SLB features by using `annotation s`.

Use an existing intranet SLB instance

You must specify three annotations. Replace "your-loadbalancer-id" with your SLB instance ID.

> **Note:**
>
> Multiple Kubernetes services can reuse the same SLB instance.

- The SLB instances created by Kubernetes through a service cannot be reused. Otherwise, the reused SLB instances may be removed incidentally. Only the SLB instances manually created in the console or created by calling API can be reused.
- The multiple services that reuse the same SLB instance cannot have the same frontend listening port. Otherwise, port conflicts will occur.
- If you reuse an SLB instance, the listener name and the virtual server group name are used to identify the SLB instance in Kubernetes. We recommend that you do not modify the listener name or virtual server group name.
- You can modify SLB instance names.
- SLB instances cannot be reused across clusters.

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
address - type : " intranet "
    service . beta . kubernetes . io / alicloud - loadbalanc  er - id
: " your - loadbalanc  er - id "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
force - override - listeners : " true "
  labels :
    run :  nginx
  name :  nginx
  namespace :  default
spec :
  ports :
 -  port :  80
    protocol :  TCP
    targetPort :  80
  selector :
    run :  nginx
  sessionAff  inity :  None
  type :  LoadBalanc  er
```

Create an HTTP-type SLB instance

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
protocol - port : " http : 80 "
  name :  nginx
  namespace :  default
spec :
  ports :
 -  port :  80
    protocol :  TCP
    targetPort :  80
  selector :
    run :  nginx
```

```
  type :  LoadBalanc  er
```

Create an HTTPS-type SLB instance

You must first create a certificate in the Alibaba Cloud console before creating an HTTPS-type SLB instance by using the following template (the certificate ID is required by the annotations in the template):

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
cert - id : " your - cert - id "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
protocol - port : " https : 443 "
  name :  nginx
  namespace :  default
spec :
  ports :
 -  port :  443
    protocol :  TCP
    targetPort :  443
  selector :
    run :  nginx
  sessionAff  inity :  None
  type :  LoadBalanc  er
```

Limit SLB instance bandwidth

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
charge - type : " paybybandw  idth "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
bandwidth : " 100 "
  name :  nginx
  namespace :  default
spec :
  ports :
 -  port :  443
    protocol :  TCP
    targetPort :  443
  selector :
    run :  nginx
  type :  LoadBalanc  er
```

Specify the SLB instance specification

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
spec : " slb . s1 . small "
  name :  nginx
  namespace :  default
```

```
spec :
  ports :
 - port : 443
   protocol : TCP
   targetPort : 443
  selector :
   run : nginx
  type : LoadBalanc er
```

Use an existing SLB instance

By default, listeners are not overridden if you use an existing SLB instance. To forcibly override the existing listeners, set `service . beta . kubernetes . io / alicloud - loadbalanc er - force - override - listeners` to `true .`

Note:

Multiple Kubernetes services can reuse the same SLB instance.

- The SLB instances created by Kubernetes through a service cannot be reused. Otherwise, the reused SLB instances may be removed incidentally. Only the SLB instances manually created in the console or created by calling API can be reused.
- The multiple services that reuse the same SLB instance cannot have the same frontend listening port. Otherwise, port conflicts will occur.
- If you reuse an SLB instance, the listener name and the virtual server group name are used to identify the SLB instance in Kubernetes. We recommend that you do not modify the listener name or virtual server group name.
- You can modify SLB instance names.
- SLB instances cannot be reused across clusters.

```
apiVersion : v1
kind : Service
metadata :
  annotation s :
   service . beta . kubernetes . io / alicloud - loadbalanc er - id
: " your_loadb alancer_id "
  name : nginx
  namespace : default
spec :
  ports :
 - port : 443
   protocol : TCP
   targetPort : 443
  selector :
   run : nginx
  type : LoadBalanc er
```

Use an existing SLB instance and forcibly override existing listeners

If you forcibly override the existing listeners, they are removed.

Note:

Multiple Kubernetes services can reuse the same SLB instance.

- The SLB instances created by Kubernetes through a service cannot be reused. Otherwise, the reused SLB instances may be removed incidentally. Only the SLB instances manually created in the console or created by calling API can be reused.
- The multiple services that reuse the same SLB instance cannot have the same frontend listening port. Otherwise, port conflicts will occur.
- If you reuse an SLB instance, the listener name and the virtual server group name are used to identify the SLB instance in Kubernetes. We recommend that you do not modify the listener name or virtual server group name.
- You can modify SLB instance names.
- SLB instances cannot be reused across clusters.

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er - id
: " your_loadb  alancer_id "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
force - override - listeners : " true "
  name :  nginx
  namespace :  default
spec :
  ports :
 -  port :  443
    protocol :  TCP
    targetPort :  443
  selector :
    run :  nginx
  type :  LoadBalanc  ere
```

Use the Worker node with specified labels as a backend server

Use a comma (,) to separate two labels, for example, K1: V1, K2: V2.

The relationship between multiple labels is `and`.

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
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 :  LoadBalanc  er
```

Set the session persistence timeout for a TCP-type SLB instance

The parameter `service . beta . kubernetes . io / alicloud - loadbalanc er - persistenc e - tim` applies only to TCP listeners.

If the SLB instance is configured with multiple TCP listener ports, this parameter setting applies to all the ports by default.

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
persistenc  e - timeout : " 1800 "
  name :  nginx
  namespace :  default
spec :
  ports :
 -  port :  443
    protocol :  TCP
    targetPort :  443
  selector :
    run :  nginx
  type :  LoadBalanc  er
```

Set session persistence for HTTP-type and HTTPS-type SLB instances (insert cookie)

Only HTTP-type and HTTPS-type SLB instances support this setting.

If an instance is configured with multiple HTTP or HTTPS listener ports, the session persistence setting applies to all the HTTP or HTTPS listener ports by default.

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
sticky - session : " on "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
sticky - session - type : " insert "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
cookie - timeout : " 1800 "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
protocol - port : " http : 80 "
  name :  nginx
  namespace :  default
spec :
  ports :
 -  port :  80
```

```
      protocol :  TCP
      targetPort :  80
    selector :
      run :  nginx
    type :  LoadBalanc  er
```

Set session persistence for HTTP-type and HTTPS-type SLB instances (server cookie)

Only HTTP-type and HTTPS-type SLB instances support this setting.

If an instance is configured with multiple HTTP or HTTPS listener ports, the session persistence setting applies to all the HTTP or HTTPS listener ports by default.

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
sticky - session : " on "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
sticky - session - type : " server "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
cooyour_co  okie : " your_cooki  e "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
protocol - port : " http : 80 "
  name :  nginx
  namespace :  default
spec :
  ports :
 -  port :  80
    protocol :  TCP
    targetPort :  80
  selector :
    run :  nginx
  type :  LoadBalanc  er
```

Specify the primary and secondary zones when creating an SLB instance

Support for the primary and secondary zones varies according to region, for example , the ap-southeast-5.

Once created, the primary and secondary zones cannot be changed.

```
apiVersion :  v1
kind :  Service
metadata :
  annotation  s :
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
master - zoneid : " ap - southeast - 5a "
    service . beta . kubernetes . io / alicloud - loadbalanc  er -
slave - zoneid : " ap - southeast - 5a "
  name :  nginx
  namespace :  default
spec :
  ports :
 -  port :  80
    protocol :  TCP
    targetPort :  80
  selector :
```

```
    run :  nginx
  type :  LoadBalanc  er
```

Use the node where the pod is located as a backend server

```
apiVersion :  v1
kind :  Service
metadata :
  name :  nginx
  namespace :  default
spec :
  externalTr  afficPolic  y :  Local
  ports :
 - port :  80
    protocol :  TCP
    targetPort :  80
  selector :
    run :  nginx
  type :  LoadBalanc  er
```

Note:

The annotations are case sensitive.

Annotation	Description	Default value
`service . beta . kubernetes . io / alicloud - loadbalanc er - protocol - port`	Use a comma (,) to separate two values, for example, https:443,http:80 .	None
`service . beta . kubernetes . io / alicloud - loadbalanc er - address - type`	Valid values: `internet` or `intranet` .	`internet`
`service . beta . kubernetes . io / alicloud - loadbalanc er - slb - network - type`	SLB instance network type. Valid values: `classic` or `vpc` .	`classic`
`service . beta . kubernetes . io / alicloud - loadbalanc er - charge - type`	Valid values: `paybytraff ic` or `paybybandw idth` .	`paybytraff ic`

Annotation	Description	Default value
`service . beta . kubernetes . io / alicloud - loadbalanc er - id`	SLB instance ID. You can specify an existing SLB instance by setting the parameter `service . beta . kubernetes . io / alicloud - loadbalanc er - id` . By default, if you specify an existing SLB instance, its original listeners will not be overridden. To override the original listeners, set the parameter `service . beta . kubernetes . io / alicloud - loadbalanc er - force - override - listeners` as `true` .	None
`service . beta . kubernetes . io / alicloud - loadbalanc er - backend - label`	Use labels to specify the Worker nodes to be mounted to the backend of the SLB instance.	None
`service . beta . kubernetes . io / alicloud - loadbalanc er - spec`	SLB instance specification. For more information, see CreateLoadBalancer.	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-persistence-timeout`	Session persistence timeout (in seconds). This parameter setting applies only to TCP listeners and the value can be `0` to `3600`. The default value is `0`, indicating that the session remains disabled. For more information, see CreateLoadBalancerTCPListener.	`0`
`service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session`	Whether to enable session persistence. Valid value: `on` | `off`. **Note:** It applies only to HTTP and HTTPS listeners. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	`off`

Annotation	Description	Default value
`service . beta . kubernetes . io / alicloud - loadbalancer - sticky - session - type`	Method used to handle the cookie. Valid values: · `insert` : Insert the cookie. · `server` : Rewrite the cookie. Note: · It applies only to HTTP and HTTPS listeners. · If you set the value of the parameter `service . beta . kubernetes . io / alicloud - loadbalancer - sticky - session` to `on` , you must specify this parameter. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-cookie-timeout`	Cookie timeout period (in seconds). Value range: `1` to `86400`. **Note:** If the `service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session` parameter is set to `on` and the `service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session-type` parameter is set to `insert`, this parameter is mandatory. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-cookie`	Cookie configured on the server. The cookie must be a string of 1 to 200 characters and can only contain ASCII letters and numeric characters. It cannot contain commas (,), semicolons (;), or spaces, and it cannot start with a dollar sign ($). **Note:** If the `service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session` parameter is set to `on` and the `service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session-type` parameter is set to `server`, this parameter is mandatory. For more information, see CreateLoadBalancerHTTPListener and CreateLoadBalancerHTTPSListener.	None

Annotation	Description	Default value
service . beta . kubernetes . io / alicloud - loadbalancer - master - zoneid	Zone ID of the primary backend server.	None
service . beta . kubernetes . io / alicloud - loadbalancer - slave - zoneid	Zone ID of the secondary backend server.	None
externalTrafficPolicy	Nodes that can be used as backend servers. Valid values: • `Cluster` : Use all backend nodes as backend servers. • `Local` : Use the nodes where pods are located as backend servers.	`Cluster`
service . beta . kubernetes . io / alicloud - loadbalancer - force - override - listeners	Determines whether to override the listeners when you specify an existing SLB instance.	`false` : Do not override.
service . beta . kubernetes . io / alicloud - loadbalancer - region	Region where the SLB instance is located.	None
service . beta . kubernetes . io / alicloud - loadbalancer - bandwidth	SLB instance bandwidth.	`50`
service . beta . kubernetes . io / alicloud - loadbalancer - cert - id	ID of a certificate on Alibaba Cloud. You must upload a certificate first.	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-flag`	Valid values: `on` | `off`.	The default value is `off`. Modifying this parameter is not required for TCP, because the health check function is enabled for TCP by default and this parameter cannot be set.
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-type`	Health check type. Valid values: `tcp` | `http`. For more information, see CreateLoadBalancerTCPListener.	`tcp`
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-uri`	URI used for health checks. Note: If the health check type is TCP, you do not need to set this parameter. For more information, see CreateLoadBalancerTCPListener.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-port`	Port used for health checks. Valid values: · `-520`: The backend port configured for the listener is used by default. · `1-65535`: The port opened on the backend server for health checks is used. For more information, see CreateLoadBalancerTCPListener.	None

Annotation	Description	Default value
service . beta . kubernetes . io / alicloud - loadbalancer - healthy - threshold	For more information, see CreateLoadBalancerTCPListener.	None
service . beta . kubernetes . io / alicloud - loadbalancer - unhealthy - threshold	The number of consecutive health check successes before the backend server is determined healthy (from failure to success). Value range: `2 to 10` For more information, see CreateLoadBalancerTCPListener.	None
service . beta . kubernetes . io / alicloud - loadbalancer - health - check - interval	Time interval between two consecutive health checks (seconds). Value range: `1 to 50` . For more information, see CreateLoadBalancerTCPListener.	None

Annotation	Description	Default value
service . beta . kubernetes . io / alicloud - loadbalancer - health - check - connect - timeout	Time period required by waiting for a health check response (in seconds). If the backend ECS instance does not send a valid response within a specified period of time, the system determines that the health check has failed. Value range: 1 to 300 . **Note:** If the value of the parameter service . beta . kubernetes . io / alicloud - loadbalancer - health - check - connect - timeout is less than the value of the parameter service . beta . kubernetes . io / alicloud - loadbalancer - health - check - interval , service . beta . kubernetes . io / alicloud - loadbalancer - health - check - connect - timeout is invalid and the timeout period equals the value of service	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout`	Time period required by waiting for a health check response (in seconds). If the backend ECS instance does not send a valid response within a specified period of time, the system determines that the health check has failed. Value range: `1` to `300`. **Note:** If the value of the parameter `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout` is less than that of the parameter `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval`, `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout` is invalid, and the timeout period equals the value of the parameter `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval`. For more information, see CreateLoadBalancerHTTPListener.	None

6.3 Support for Ingress

In Kubernetes clusters, Ingress is a collection of rules that authorize inbound connection to the cluster services and provides you with Layer-7 Server Load Balancer capabilities. You can provide the Ingress configuration with externally accessible URL, Server Load Balancer, SSL, and name-based virtual host.

Prerequisites

To test the complex routing service, create an Nginx application in this example. You must create the Nginx deployment and multiple services in advance to observe the routing effect. Replace with your own service in the actual test.In the actual test enter your own service.

```
root @ master  #  kubectl   run   nginx  -- image = registry . cn -
hangzhou . aliyuncs . com / acs / netdia : latest

root @ master  #  kubectl   expose   deploy   nginx  -- name = http -
svc  -- port = 80  -- target - port = 80
root @ master  #  kubectl   expose   deploy   nginx  -- name = http -
svc1  -- port = 80  -- target - port = 80
root @ master  #  kubectl   expose   deploy   nginx  -- name = http -
svc2  -- port = 80  -- target - port = 80
root @ master  #  kubectl   expose   deploy   nginx  -- name = http -
svc3  -- port = 80  -- target - port = 80
```

Simple routing service

Create a simple Ingress service by using the following commands. All the accesses to the `/ svc` path are routed to the Nginx service. `nginx . ingress . kubernetes . io / rewrite - target : /` redirects the path `/ svc` to the path / that can be recognized by backend services.

```
root @ master  #  cat  << EOF  |  kubectl   create  - f  -
apiVersion :  extensions / v1beta1
kind :  Ingress
metadata :
  name :  simple
  annotation  s :
    nginx . ingress . kubernetes . io / rewrite - target : /$ 2
spec :
  rules :
 -  http :
      paths :
     -  path : / svc (/|$)(.*)
        backend :
          serviceNam  e :  http - svc
          servicePor  t :  80
EOF

root @ master  #  kubectl   get   ing
NAME              HOSTS           ADDRESS              PORTS       AGE
```

```
simple              *               101 . 37 . 192 . 211     80
11s
```

Now visit `http :// 101 . 37 . 192 . 211 / svc` to access the Nginx service.

Simple fanout routing based on domain names

If you have multiple domain names providing different external services, you can generate the following configuration to implement a simple fanout effect based on domain names:

```
root @ master  #  cat  << EOF  |  kubectl   create  - f  -
apiVersion :  extensions / v1beta1
kind :  Ingress
metadata :
  name :  simple - fanout
spec :
  rules :
 -  host :  foo . bar . com
    http :
      paths :
     -  path : / foo
        backend :
          serviceNam  e :  http - svc1
          servicePor  t :  80
     -  path : / bar
        backend :
          serviceNam  e :  http - svc2
          servicePor  t :  80
 -  host :  foo . example . com
    http :
      paths :
     -  path : / film
        backend :
          serviceNam  e :  http - svc3
          servicePor  t :  80
EOF
root @ master  #  kubectl   get   ing
NAME              HOSTS            ADDRESS            PORTS      AGE
simple - fanout    *               101 . 37 . 192 . 211     80
  11s
```

Then, you can access the `http - svc1` service by using `http :// foo . bar . com / foo`, access the `http - svc2` service by using `http :// foo . bar . com / bar`, and access the `http - svc3` service by using `http :// foo . example . com / film`.

Note:

- In a production environment, point the domain name to the preceding returned address `101 . 37 . 192 . 211`.

- - In a testing environment, you can modify the `hosts` file to add a domain name mapping rule.

```
101 . 37 . 192 . 211   foo . bar . com
101 . 37 . 192 . 211   foo . example . com
```

Default domain name of simple routing

It does not matter if you do not have the domain name address. Container Service binds a default domain name for Ingress service. You can use this default domain name to access the services. The domain name is in the format of `*.[ cluster - id ].[ region - id ]. alicontain er . com`. You can obtain the address on the cluster Basic Information page in the console.

Use the following configuration to expose two services with the default domain name.

```
root @ master  #  cat  << EOF  |  kubectl   create  - f  -
apiVersion :  extensions / v1beta1
kind :  Ingress
metadata :
  name :  shared - dns
spec :
  rules :
 -  host :  foo .[ cluster - id ].[ region - id ]. alicontain  er .
com   ## Replace   with   the   default   service   access   domain
name   of   your   cluster .
    http :
      paths :
     -  path : /
        backend :
          serviceNam  e :  http - svc1
          servicePor  t :  80
 -  host :  bar .[ cluster - id ].[ region - id ]. alicontain  er .
com   ## Replace   with   the   default   service   access   domain
name   of   your   cluster .
    http :
      paths :
     -  path : /
        backend :
          serviceNam  e :  http - svc2
          servicePor  t :  80
EOF
root @ master  #  kubectl   get   ing
NAME              HOSTS           ADDRESS             PORTS      AGE
shared - dns     foo .[ cluster - id ].[ region - id ]. alicontain
er . com , bar .[ cluster - id ].[ region - id ]. alicontain  er .
com              47 . 95 . 160 . 171     80          40m
```

Then, you can access the `http - svc1` service by using `http :// foo .[ cluster - id ].[ region - id ]. alicontain  er . com /`and access the `http - svc2` service by using `http :// bar .[ cluster - id ].[ region - id ]. alicontain  er . com`.

Configure a safe routing service

Management of multiple certificates is supported to provide security protection for your services.

1. Prepare your service certificate.

   If no certificate is available, generate a test certificate in the following method:

   > **Note:**
   > The domain name must be consistent with your Ingress configuration.

   ```
   root @ master  #  openssl   req  - x509  - nodes  - days   365
   - newkey   rsa : 2048  - keyout   tls . key  - out   tls . crt  -
   subj  "/ CN = foo . bar . com / O = foo . bar . com "
   ```

   The above command generates a certificate file *tls . crt* and a private key file *tls . key* .

   Create a Kubernetes secret named *foo . bar* using the certificate and private key. The secret must be referenced when you create the Ingress.

   ```
   root @ master  #  kubectl   create   secret   tls   foo . bar  --
   key   tls . key  -- cert   tls . crt
   ```

2. Create a safe Ingress service.

   ```
   root @ master  #  cat  << EOF  |  kubectl   create  - f  -
   apiVersion :  extensions / v1beta1
   kind :  Ingress
   metadata :
     name :  tls - fanout
   spec :
     tls :
    -  hosts :
      -  foo . bar . com
       secretName :  foo . bar
     rules :
    -  host :  foo . bar . com
       http :
         paths :
        -  path : / foo
           backend :
             serviceNam  e :  http - svc1
             servicePor  t :  80
        -  path : / bar
           backend :
             serviceNam  e :  http - svc2
             servicePor  t :  80
   EOF
   root @ master  #  kubectl   get   ing
   NAME                 HOSTS           ADDRESS            PORTS
   AGE
   ```

```
tls - fanout        *                 101 . 37 . 192 . 211      80
   11s
```

3. Follow the notes in Simple fanout routing based on domain names to configure the `hosts` file or set the domain name to access the TLS service.

   You can access the `http - svc1` service by using `http :// foo . bar . com / foo` and access the `http - svc2` service by using `http :// foo . bar . com / bar`.

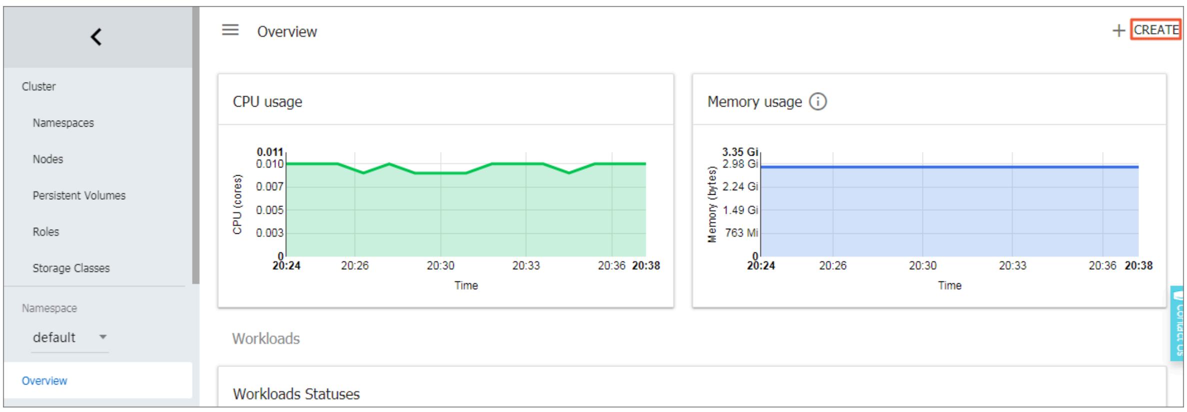
   You can also access the HTTPS service by using HTTP. By default, Ingress redirects HTTP access configured with HTTPS to the HTTPS address. Therefore, access to `http :// foo . bar . com / foo` will be automatically redirected to `https :// foo . bar . com / foo`.

Deploy Ingress in Kubernetes dashboard

1. 1. Save the following yml code to the `nginx - ingress . yml` file.

```
apiVersion :  extensions / v1beta1
kind :  Ingress
metadata :
  name :  simple
spec :
  rules :
  -  http :
      paths :
     -  path : / svc
        backend :
          serviceNam  e :  http - svc
          servicePor  t :  80
```

2. Log on to the Container Service console. In the left-side navigation pane under Kubernetes, click Clusters. Then click Dashboardon the right of the target cluster.
3. Click CREATE in the upper-right corner to create an application.

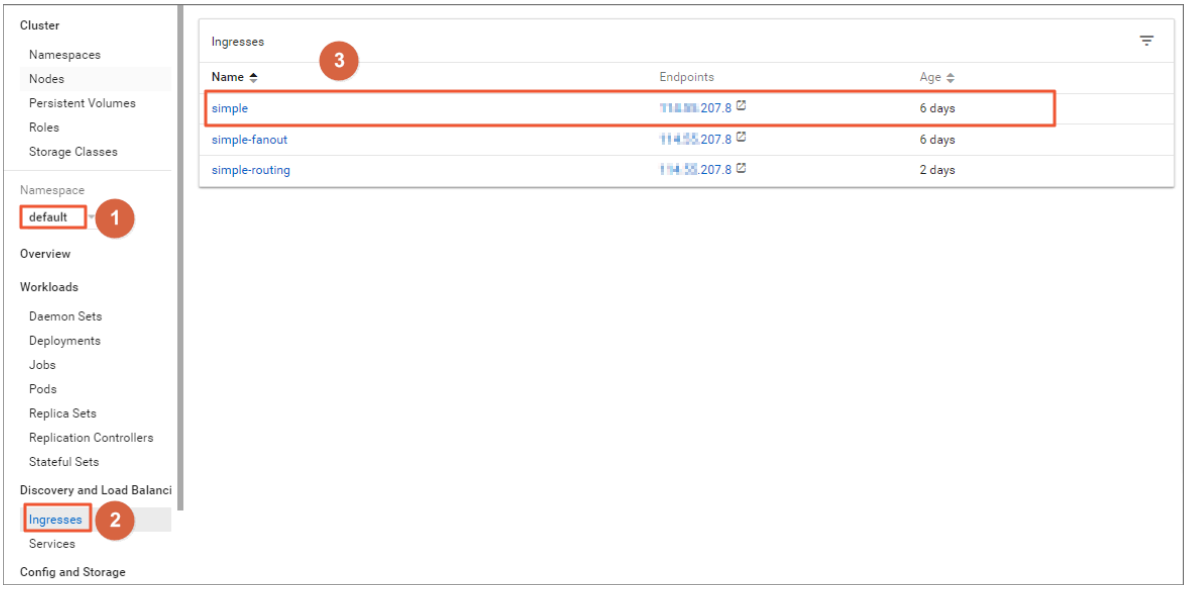


4. Click the CREATE FROM FILE tab. Select the `nginx - ingress . yml` file you saved.

5. Click UPLOAD.

   Then an Ingress Layer-7 proxy route will be created to the `http - svc` service.
6. Click default under Namespace in the left-side navigation pane. Click Ingresses in the left-side navigation pane.

   You can view the created Ingress resource and its access address `http :// 118 . 178 . 174 . 161 / svc` .



7. Enter the address in the browser to access the created `http - svc` service.

6.4 Analyze logs of Ingress to monitor access to Ingress

This topic describes how to enable Ingress to collect logs, how to analyze log reports of different types, and how to use reports by using alarm configurations and report subscription. Alibaba Cloud Container Service for Kubernetes (ACK) provides Ingress to collect all HTTP requests to a standard output. You can use Log Service (integrated with ACK) to create dashboards to analyze logs of Ingress and monitor access to Ingress.

Before you begin

1. Install the log component in a Kubernetes cluster.

   > Note:
   >
   > If you want to create a new Kubernetes cluster oruse an existing Kubernetes cluster in which the log component is not installed,follow these steps:

- For information about how to install the log component in a Kubernetes cluster when the cluster is created, see Create a Kubernetes cluster.
- For information about how to install the log component in a Kubernetes cluster after the cluster is created, see Use Log Service to collect Kubernetes cluster logs.

2. Upgrade the log component `alibaba-log-controller` of the target Kubernetes cluster.

   The log component `alibaba-log-controller` is a deployment application located in the kube-system namespace of the target Kubernetes cluster. To upgrade it, you must modify the following two parameters:

   - `Image name`: Replace `{region-id}` in the image name *registry-vpc.{region-id}.aliyuncs.com/acs/log-controller* with the ID of the region to which the target Kubernetes cluster belongs. For example, `{region-id}` can be replaced with cn-hangzhou, cn-beijing, or ap-southeast-1.
   - `Image version`: It must be the version `0.2.0.0-76648ee-aliyun` or later.

   You can choose one of the following two upgrade methods:

   - Run the `kubectl edit deployment alibaba-log-controller -n kube-system` command.
   - Update by using the Container Service console.

     To do so, follow these steps:

     a. Log on to the Container Service console.
     b. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments.
     c. Select the target Kubernetes cluster and the kube-system namespace, find alibaba-log-controller, and then, in the Action column, click Edit.

Deploy the configurations for Ingress to collect logs

Overview

The following are the configurations for Ingress to collect logs. The configurations can be viewed as the expanded Kubernetes Custom Resource Definitions (CRDs).

Therefore, the configurations are referred to as only CRD configurations later in this topic. When the CRD configurations are deployed, the log component automatically creates the parameters and report resources that are related to Log Service.

```
apiVersion :  log . alibabaclo  ud . com / v1alpha1
kind :  AliyunLogC  onfig
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
  productCod  e :  k8s - nginx - ingress
 #  logtail   config   detail
  logtailCon  fig :
    inputType :  plugin
   #  logtail   config   name ,  should   be   same   with  [
metadata . name ]
    configName :  k8s - nginx - ingress
    inputDetai  l :
      plugin :
        inputs :
       -  type :  service_do  cker_stdou  t
          detail :
            IncludeLab  el :
              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_use  r
           -  time
           -  method
           -  url
           -  version
           -  status
           -  body_bytes  _sent
           -  http_refer  er
           -  http_user_  agent
           -  request_le  ngth
           -  request_ti  me
           -  proxy_upst  ream_name
           -  upstream_a  ddr
           -  upstream_r  esponse_le  ngth
           -  upstream_r  esponse_ti  me
           -  upstream_s  tatus
           -  req_id
           -  host
            NoKeyError :  true
            NoMatchErr  or :  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 *).*
            SourceKey :  content
```

- If you have deployed the CRD configurations before you upgrade the log component alibaba-log-controller to version `0 . 2 . 0 . 0 - 76648ee - aliyun`, you must first delete the deployed CRD configurations, and then redeploy the CRD configurations after the upgrade.
- The preceding CRD configurations take effect only for the log format of the default Ingress Controller of ACK. If the log format is modified, you must modify the regular expression (the `processor_ regex` part) in the CRD configurations according to the procedures described inConfigure Kubernetes log collection on CRD.

Procedure

You can use one of the following two methods to deploy the CRD configurations:

- Use a kubectl command.

  Save the preceding CRD configurations as an *nginx - ingress . yaml* file, and then run the `kubectl apply - f` command.
- Use an orchestration template.
  1. Log on to the Container Service console.
  2. Save the preceding CRD configurations as an orchestration template. For more information, see Create an orchestration template.
  3. In the default namespace of the target Kubernetes cluster, use this template to create an application.

View logs and reports of Ingress

1. Log on to the Log Service console
2. Click the Project associated with the target Kubernetes cluster.

   > Note:
   > The default name of the Project is k8s-log-{cluster-id}.

   Then, the nginx-ingress Logstore is display. All the logs of Ingress are store in this Logstore.

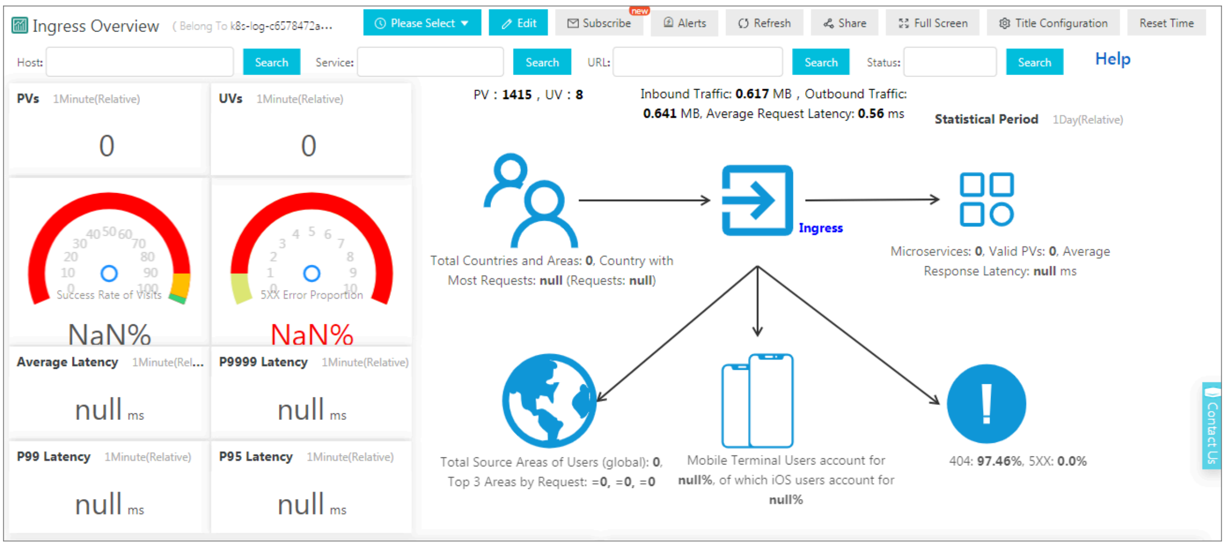
3. In the left-side navigation pane, click Dashboard to view all the reports of Ingress.

> **Note:**
> The following are the reports of Ingress logs: Ingress Overview , Ingress Access Center, Ingress Monitoring Center, Ingress Monitoring Center for Blue/Green Deployment, and Ingress Exceptions Center.

Ingress Overview report

An Ingress Overview report displays the overall status of the Ingress with the following information:

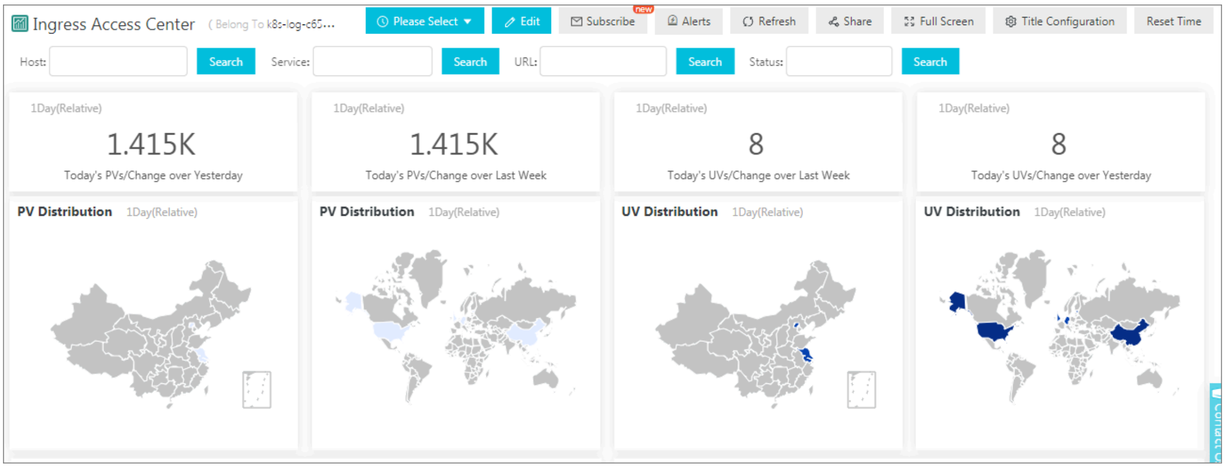
- `Overall status ( each day )`: PV, UV, traffic, request latency, and the ratio of the number of mobile terminal users to the number of all users.
- `Website status in real time ( each minute )`: PV, UV, rate of successful access, the proportion of 5XX errors in all errors, average latency, P95 latency, and P99 latency.
- `Statistics of users requests ( each day )`: PVs of today and seven days, visit distribution by area, the top 10 province and cities by requests, shares of mobile terminals, and shares of Android and IOS terminals.
- Statistics of Top URLs (each hour): Top 10 URLs by request, Top10 URLs by latency, Top 10 5XX URLs and Top 10 404 URLs.



Ingress Access Center

Ingress Access Center provides the statistics of access requests that can be used to analyze the operation status. This report contains the following information:

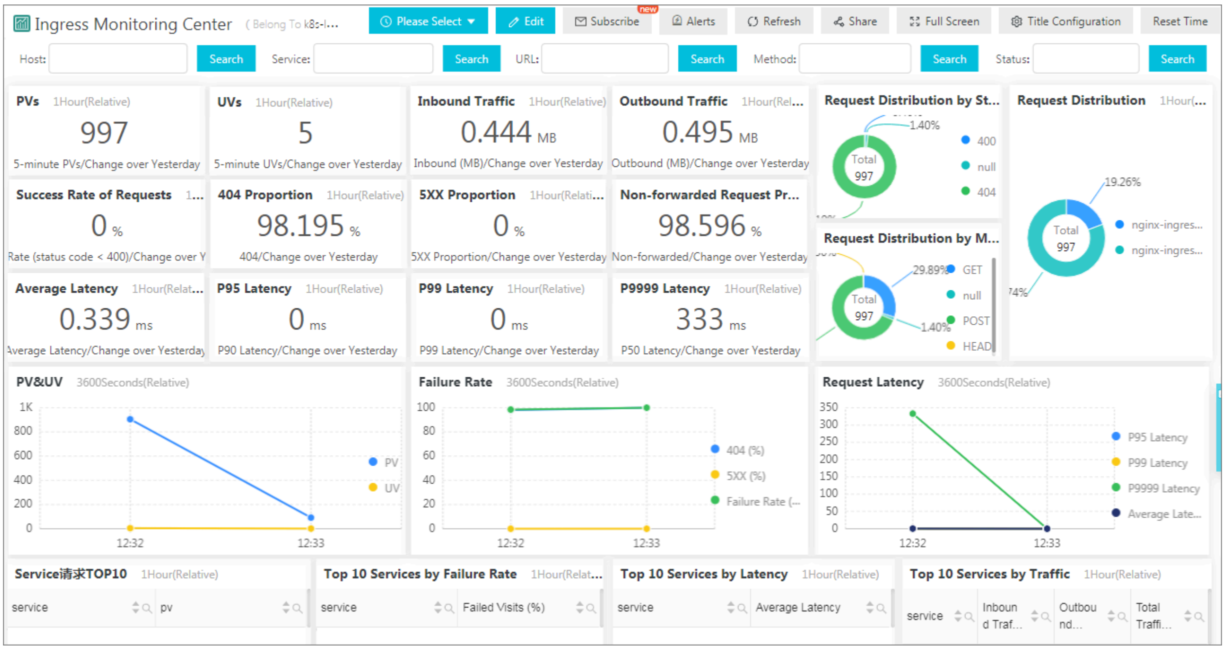
- UV and PV of the current day
- UV and PV distribution
- UV and PV trend
- Top 10 areas and cities by request
- Top browsers
- Top IP addresses of accesses
- Shares of mobile terminals
- Shares of Android and IOS



Ingress Monitoring Center

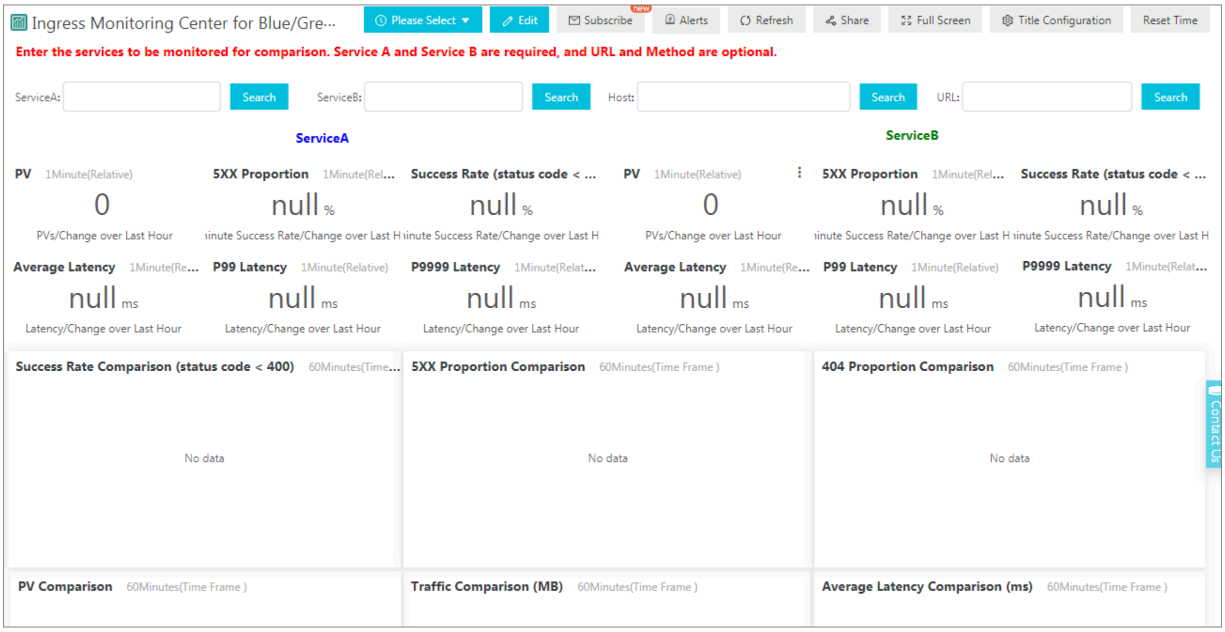
Ingress Monitoring Center provides the real-time monitoring statistics for a website. This report contains the following statistics:

- Rate of successful requests
- Proposition of 404 status codes
- Proposition of status codes of 5XX
- Non-forwarded request proportion
- Average latency
- P95, P99, and P9999 latency
- Top 10 services by requests
- Top 10 services by failures
- Top 10 services by latency
- Top 10 services by traffic



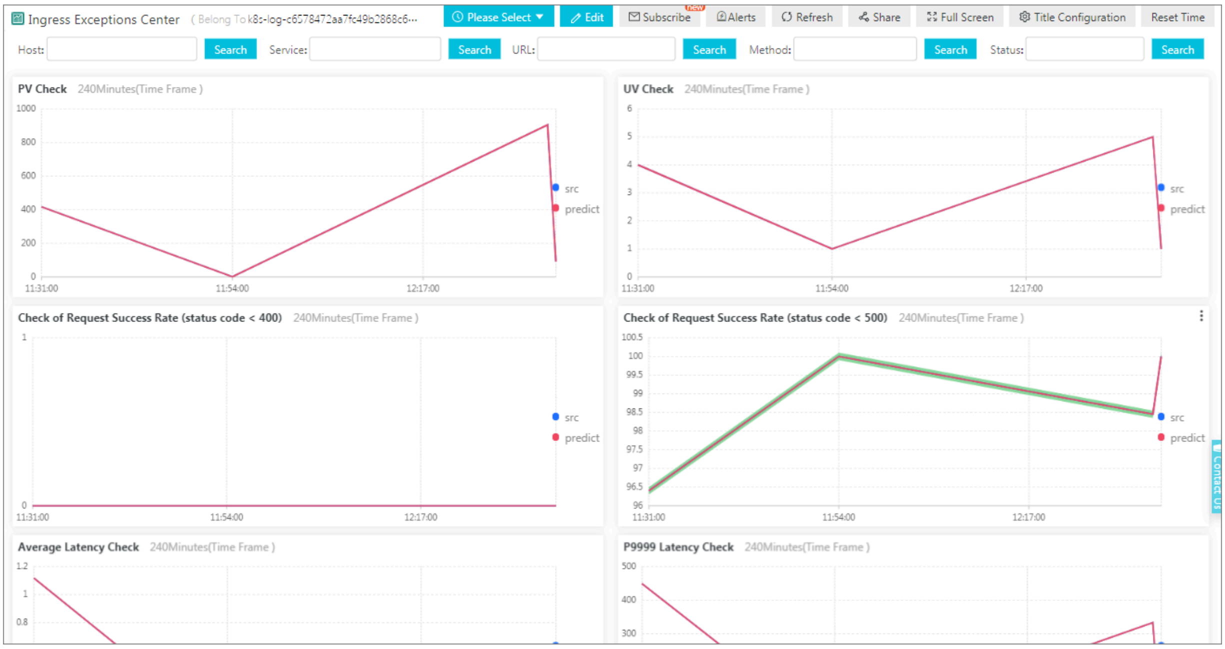
Ingress Monitoring Center for Blue/Green Deployment

This report is used to monitor version releases and compare the blue and green versions. This helps you quickly find release exceptions and then roll back the the original version. In this report, you need to select the blue and green versions (for example, Service A and Service B). The report dynamically displays the statistics about the two versions, including PV, the ratio of 5XX errors to all errors, rate of success, average latency, traffic, and latency of P5, P99, and P9999.



Ingress Exceptions Center

Ingress Exceptions Center operates on the basis of the algorithm of machine learning provided by Log Service. It automatically detects exceptions from the Ingress metrics by using multiple timing analysis algorithms.



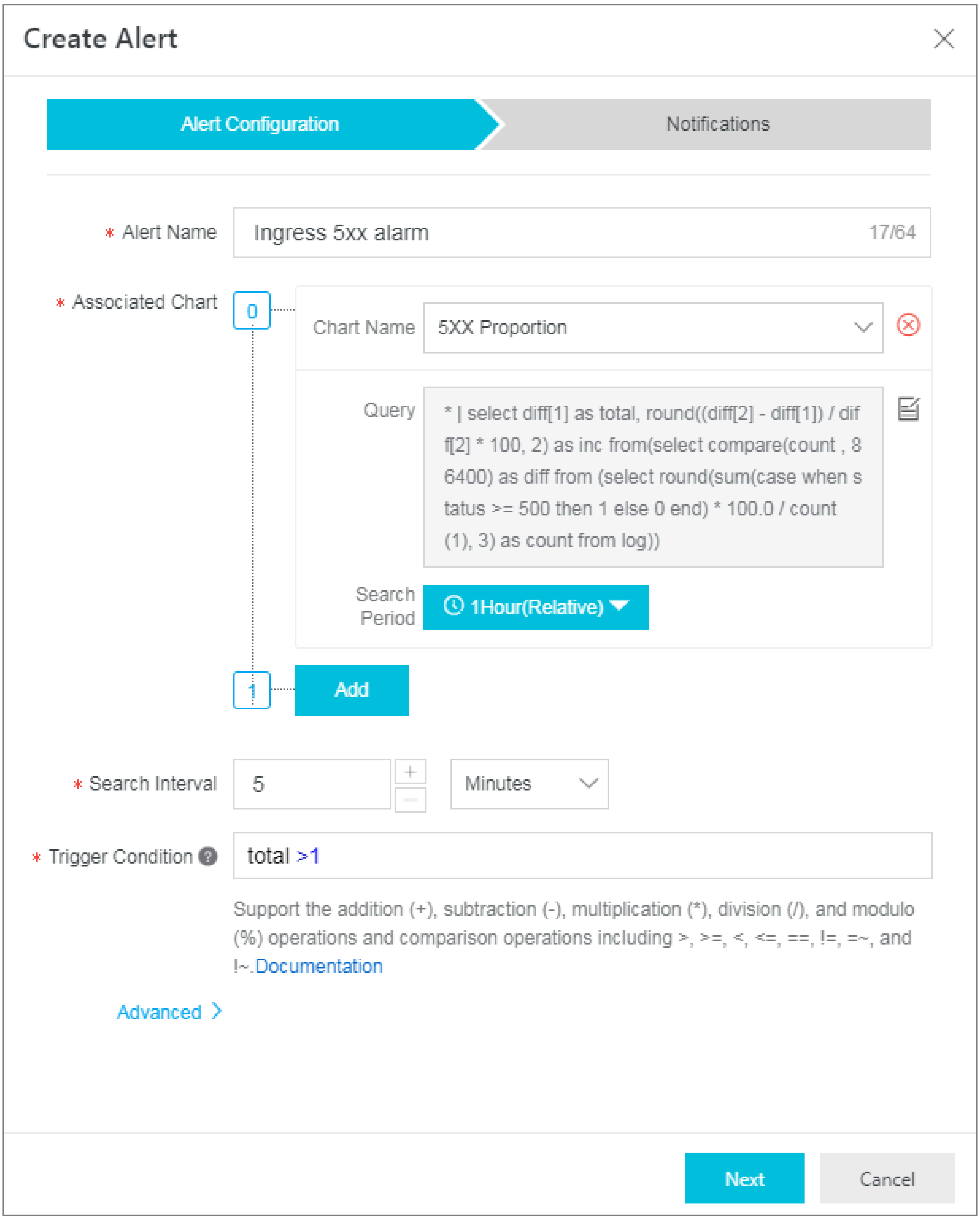
Configure an alarm

You can configure an alarm for each of the preceding reports. For more information, see Configure an alarm.

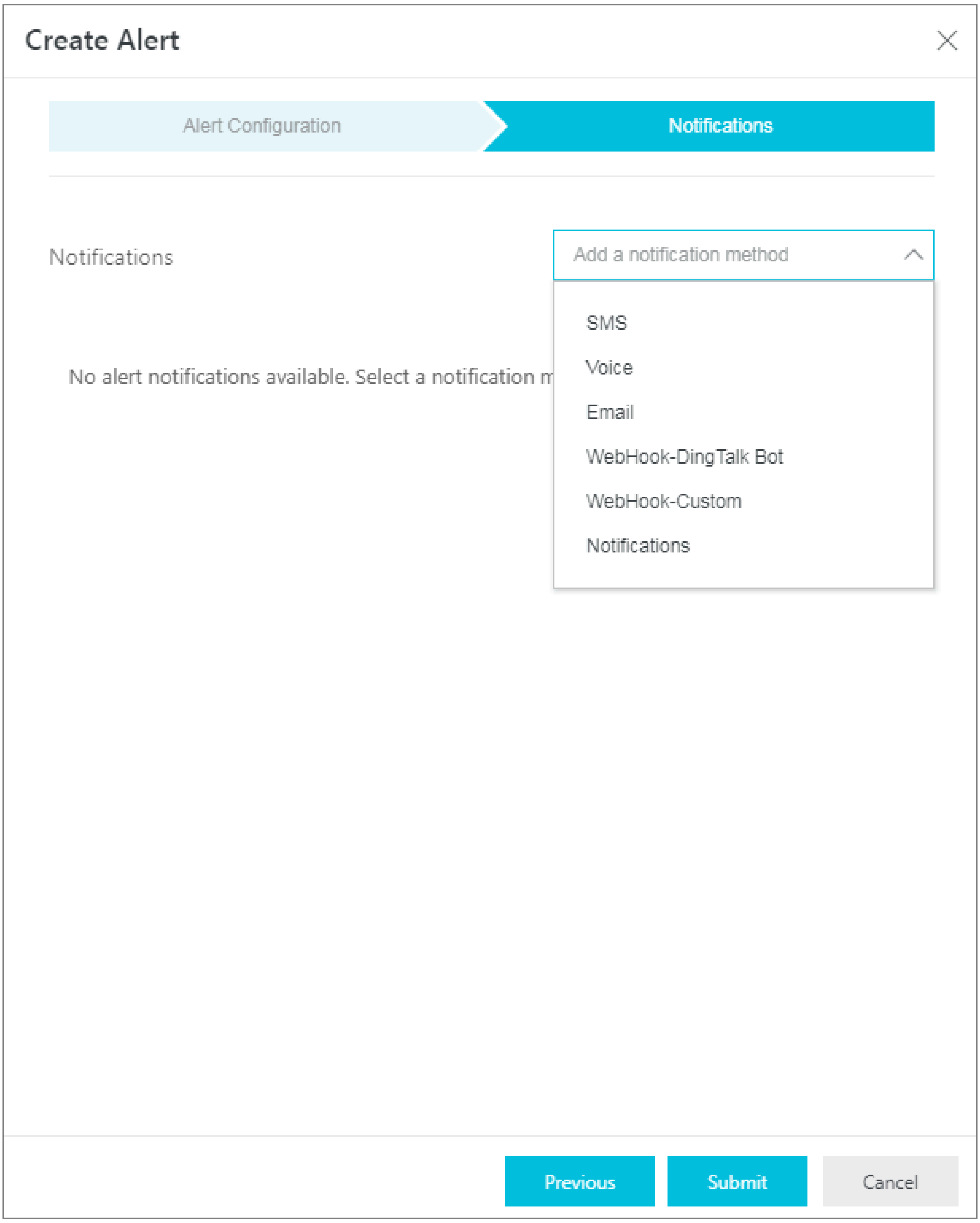
The following shows how to configure an alarm for the statistics of the porportion of 5xx status codes:

1. Open the Ingress Monitoring Center report, move your pointer to the upper-right corner of the 5XX Proportion chart, and then, in the displayed box, click Create Alarm.

2. Set the alarm name, search interval, and enter a trigger condition `total > 1`.



3. Set a notification method as needed.



Subscribe to a scheduled report

You can schedule Log Service to render a report to a figure and then send the figure to you through an email or sent the figure to a specific DingTalk group. For more information, see Subscribe to dashboard snapshots.

The following shows how to subscribe to a scheduled Ingress overview report (the figure generated by rendering the report is sent to the specified DingTalk group at 10:00 every day):

1. Open the Ingress overview report. Then, in the upper-right corner, click Subscribe.
2. On the displayed page, select `Daily` and `10 : 00` from the two drop-down lists of Frequency, turn off Add Watermark.
3. Select the WebHook-DingTalk Bot from the Notifications drop-down list, and then enter the request URL.

6.5 Ingress configurations

Alibaba Cloud Container Service provides the highly reliable Ingress controller components and integrates with Alibaba Cloud Server Load Balancer to provide the flexible and reliable Ingress service for your Kubernetes clusters.

See the following Ingress orchestration example. You must configure the annotation when creating an Ingress in the Container Service console. Some configurations must create dependencies. For more information, see Create an Ingress in the Container Service console, Support for Ingress, and Kubernetes Ingress. Ingress also supports the configuration of configmap. For more information, see https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/.

```
apiVersion :  extensions / v1beta1
kind :  Ingress
metadata :
  annotation  s :
    nginx . ingress . kubernetes . io / service - match : ' new -
nginx :  header (" foo ", /^ bar $/)'           # Gray   release
rule . Header   is   used   in   this   example .
    nginx . ingress . kubernetes . io / service - weight : ' new -
nginx :  50 , old - nginx :  50 '                # Traffic   weight
annotation
  creationTi  mestamp :  null
  generation :  1
  name :  nginx - ingress
  selfLink : / apis / extensions / v1beta1 / namespaces / default /
ingresses / nginx - ingress
spec :
  rules :                                                       ##
Ingress   rule
 - host :  foo . bar . com
    http :
      paths :
     -  backend :
          serviceNam  e :  new - nginx
          servicePor  t :  80
        path : /
     -  backend :
```

```
            serviceNam  e :  old - nginx
            servicePor  t :  80
          path : /
  tls :                                                          ##
  Enable   TLS   to   set   a   secure   Ingress .
   -  hosts :
     - *. xxxxxx . cn - hangzhou . alicontain  er . com
     -  foo . bar . com
      secretName :  nginx - ingress - secret
  ## Secret   name
  status :
    loadBalanc  er : {}
```

Annotation

You can configure an ingress annotation, specifying the ingress controller to use, rules for routing, such as routing weight rules, grayscale publish, and rewrite rules. For more information, see https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/.

For example, a typical rewrite annotation `nginx . ingress . kubernetes . io / rewrite - target : /` redirects the path / *path* to the path / that can be recognized by the backend services.

Rules

The rules indicate those that authorize the inbound access to the cluster and are generally the HTTP rules, including the domain name (virtual hostname), URL access path, service name, and port.

You must complete the following configurations for each HTTP rule:

- Host: Enter the testing domain name of an Alibaba Cloud Kubernetes cluster or a virtual hostname, such as `foo . bar . com`.
- Path: Specify the URL path of the service access. Each path is associated with a backend service. Before Alibaba Cloud Server Load Balancer forwards the traffic to the backend, all inbound requests must match with the domain name and path.

- Backend configuration: Service configuration that is a combination of `service : port` and traffic weight. The Ingress traffic is forwarded to the matched backend services based on the traffic weight.
  - Name: The name of the backend service forwarded by Ingress.
  - Port: The port exposed by the service.
  - Weight: The weight rate of each service in a service group.

  Note:

  1. The service weight is calculated in relative values. For example, if both service weights are set to 50, the weight ratio of both services is 50%.
  2. A service group (a service with the same Host and Path in the same ingress yaml) has a default weight value of 100 and the weight is not explicitly set.

Grayscale publish

Container Service supports different traffic segmentation methods for grayscale publish and AB test scenarios.

Note:
Currently, the Alibaba Cloud Container Service Kubernetes Ingress Controller requires `0 . 12 . 0 - 5` and above to support the traffic segmentation feature.

1. Traffic segmentation based on the request header.
2. Traffic segmentation based on cookie.
3. Traffic segmentation based on query (request) parameters.

After the grayscale rule is configured, the request that matches the grayscale publish rule can be routed to the set service. If the service sets a weight rate of less than 100%, requests that match the grayscale publish rule continue to be routed to the corresponding service based on the weight rate.

TLS

You can encrypt the Ingress by specifying a secret that contains the TLS private key and certificate to implement the secure Ingress access. The TLS secret must contain the certificate named tls.crt and private key named tls.key. For more information about the TLS principles, see TLS. For how to create a secret, see Configure a safe routing service.

Label

You can add tags for Ingress to indicate the characteristics of the Ingress.

6.6 Create an Ingress in the Container Service console

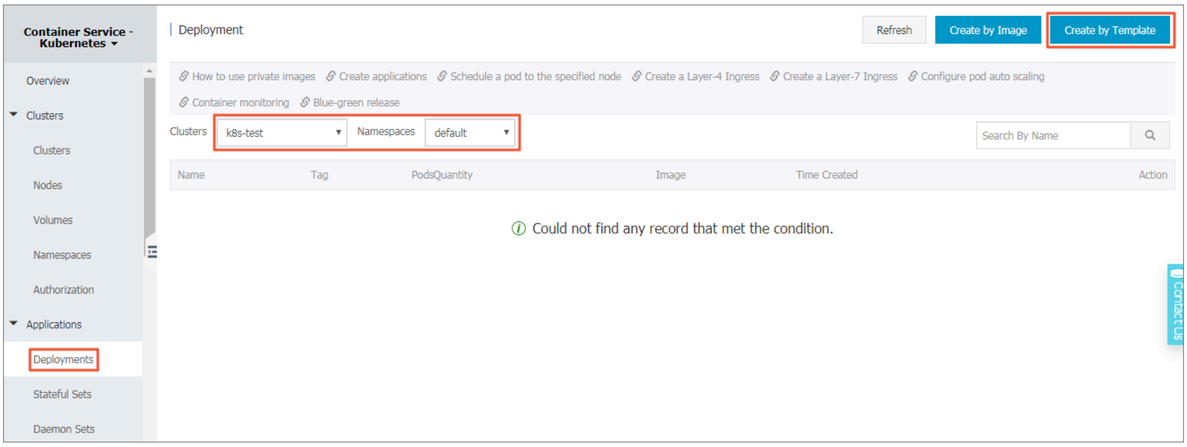
Alibaba Cloud Container Service console integrates with the Ingress service, which allows you to quickly create an Ingress service in the Container Service console to build the flexible and reliable traffic access layer.

Prerequisites

- You have successfully created a Kubernetes cluster and Ingress controller is running normally in the cluster. For how to create a Kubernetes cluster, see Create a Kubernetes cluster.
- Log on to the master node by using SSH. For more information, see Access Kubernetes clusters by using SSH.

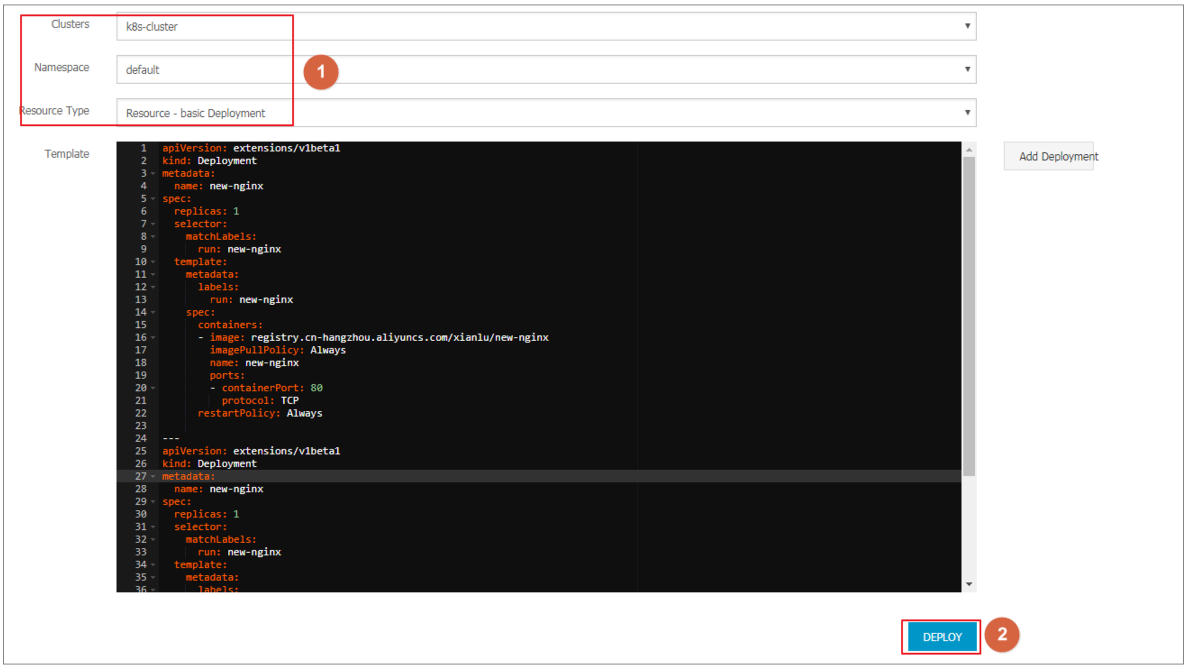
Step 1: Create a deployment and a service

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments.
3. Click Create by template in the upper-right corner.



4. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click DEPLOY.

   In this example, three nginx applications are created. One for the old application (old-nginx), one for the new (new-nginx), and an application for testing the cluster access domain name (domain-nginx).



The orchestration template for old-nginx is as follows:

```
apiVersion :  extensions / v1beta1
kind :  Deployment
metadata :
  name :  old - nginx
spec :
  replicas :  2
  selector :
    matchLabel  s :
      run :  old - nginx
  template :
    metadata :
      labels :
        run :  old - nginx
    spec :
      containers :
     - image :  registry . cn - hangzhou . aliyuncs . com / xianlu
/ old - nginx
        imagePullP  olicy :  Always
        name :  old - nginx
        ports :
       - containerP  ort :  80
          protocol :  TCP
      restartPol  icy :  Always
---
apiVersion :  v1
kind :  Service
```

```
metadata :
  name :  old - nginx
spec :
  ports :
 -  port :  80
    protocol :  TCP
    targetPort :  80
  selector :
    run :  old - nginx
  sessionAff  inity :  None
  type :  NodePort
```

The orchestration template for new-nginx is as follows:

```
apiVersion :  extensions / v1beta1
kind :  Deployment
metadata :
  name :  new - nginx
spec :
  replicas :  1
  selector :
    matchLabel  s :
      run :  new - nginx
  template :
    metadata :
      labels :
        run :  new - nginx
    spec :
      containers :
     -  image :  registry . cn - hangzhou . aliyuncs . com / xianlu
/ new - nginx
        imagePullP  olicy :  Always
        name :  new - nginx
        ports :
       -  containerP  ort :  80
          protocol :  TCP
      restartPol  icy :  Always
---
apiVersion :  v1
kind :  Service
metadata :
  name :  new - nginx
spec :
  ports :
 -  port :  80
    protocol :  TCP
    targetPort :  80
  selector :
    run :  new - nginx
  sessionAff  inity :  None
  type :  NodePort
```

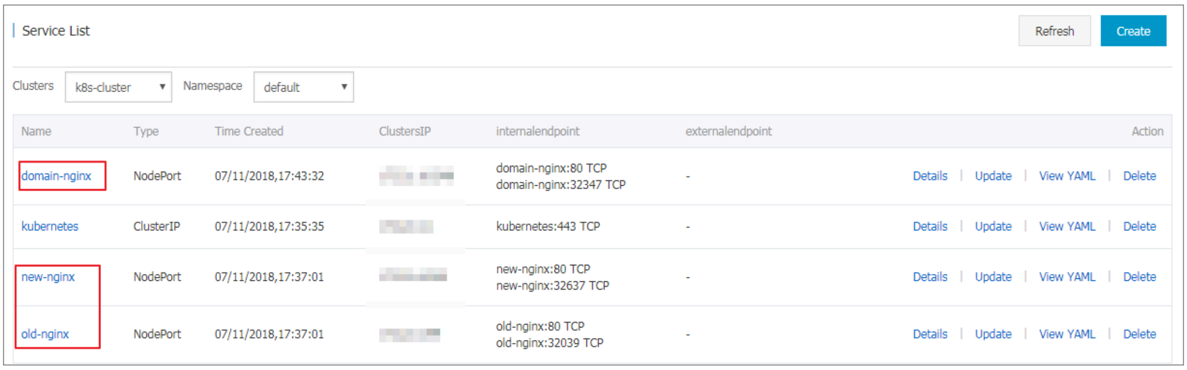
The orchestration template for domain-nginx is as follows:

```
apiVersion :  apps / v1beta2  #  For   versions   before   1 . 8 .
0   use   apps / v1beta1
kind :  Deployment
metadata :
  name :  domain - nginx
  labels :
    app :  nginx
spec :
```

```
    replicas :  2
    selector :
      matchLabel  s :
        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 :
         - containerP  ort :  80

 ---
  apiVersion :  v1
  kind :  Service
  metadata :
    name :  domain - nginx
  spec :
    ports :
   - port :  80
      protocol :  TCP
      targetPort :  80
    selector :
      app :  nginx
    sessionAff  inity :  None
    type :  NodePort
```

5. In the left-side navigation pane under Container Service-Kubernetes, choose Discovery and Load Balancing > Services.

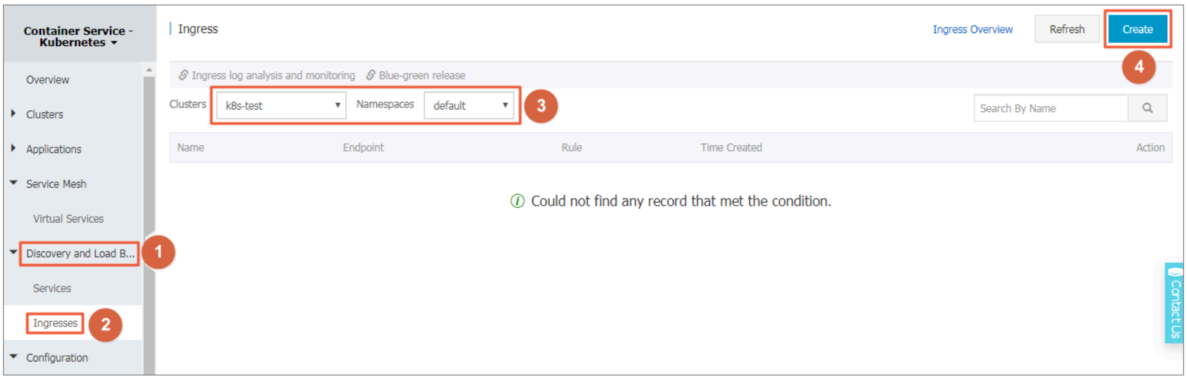
   After the service is created, you can see it on the Service List page.



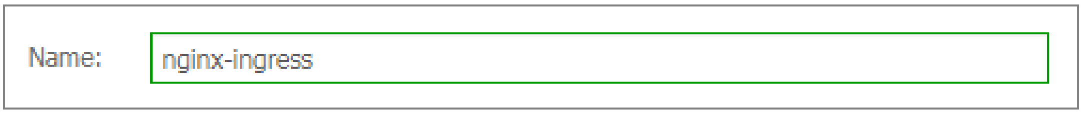
Step 2: Create an Ingress

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Discovery and Load Balancing > Ingresses.

3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Then click Create in the upper-right corner.



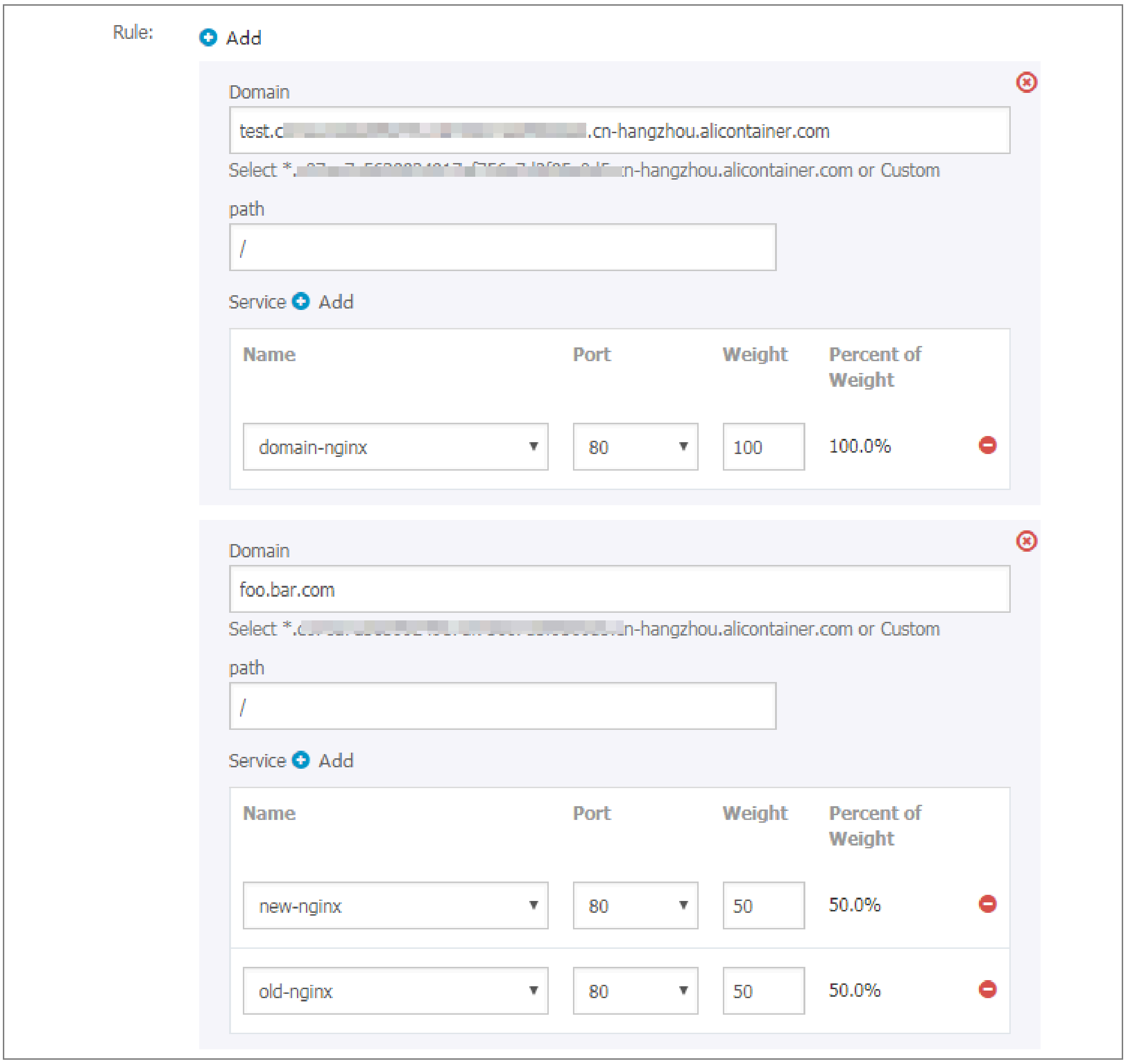
4. In the displayed dialog box, enter the Ingress name. In this example, enter nginx-ingress.



5. Configure the rules.

   The Ingress rules are the rules that authorize the inbound access to the cluster and are generally the HTTP rules. Configure the domain name (virtual hostname), URL path, service name, and port. For more information, see Ingress configurations.

   In this example, add a complicated Ingress rule. Configure the default test domain name and virtual hostname of the cluster to display the Ingress service based on the domain names.



   - The simple Ingress based on the default domain name, that is, provide the access service externally by using the default domain name of the cluster.
     - Domain: Enter the default domain name of the cluster. In this example, use `test .[ cluster - id ].[ region - id ]. alicontain er . com` .

The default domain name of this cluster is displayed in the Create dialog box, in the `*.[ cluster - id ].[ region - id ]. alicontain er . com` format. You can also obtain the default domain name on the Basic Information page of the cluster.

- Service: Configure the access path, name, and port of the service.
  - Path: Specify the URL path of the service access. The default is the root path `/`, which is not configured in this example. Each path is associated with a backend service. Before Alibaba Cloud Server Load Balancer forwards the traffic to the backend, all inbound requests must match with the domain name and path.
  - Service configuration: The backend configuration, which is a combination of service name, port, and service weight. The configuration of multiple services in the same access path is supported, and Ingress traffic is split and is forwarded to the matched backend services.

· The simple fanout Ingress based on the domain name. In this example, use a virtual hostname as the testing domain name to provide the access service externally. You can use the recorded domain name in the production environment to provide the access service. You can use the recorded domain name in the production environment to provide the access service.

- Domain: In this example, use the testing domain name `foo . bar . com`.

  You must modify the hosts file to add a domain name mapping rule.

```
118 . 178 . 108 . 143   foo . bar . com  #   Ingress   IP
address
```

- Service: Configure the access path, name, and port of the service.
  - Path: Specify the URL path of the service access. Path is not configured in this example, and the root path is `/`.
  - Name: In this example, set up both new and old services, nginx-new and nginx-old.
  - Port: Expose 80 port.
  - Weight settings: Set the weight of multiple services under this path. The service weight is calculated by relative value. The default value is 100. As shown in this example, the service weight values of both the old and new versions are 50, which means that the weight rate of both services is 50%.

6. Grayscale publish configuration.

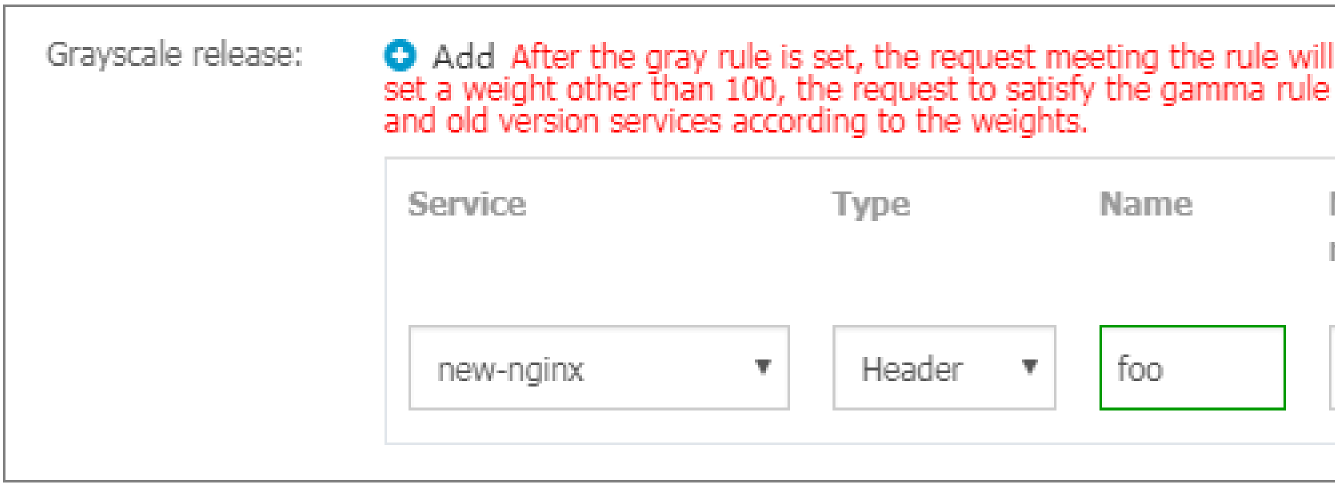
> **Note:**
> Currently, the Alibaba Cloud Container Service Kubernetes Ingress Controller requires `0.12.0-5` and above to support the traffic segmentation feature.

Container Service supports different traffic segmentation methods for grayscale publish and AB test scenarios.

a. Traffic segmentation based on the request header.
b. Traffic segmentation based on cookie.
c. Traffic segmentation based on query (request) parameters.

After the grayscale rule is configured, the request that matches the grayscale publish rule can be routed to the new service version new-nginx. If the service sets a weight rate of less than 100%, requests that match the grayscale publish rule continue to be routed to the corresponding service based on the weight rate.

In this case, set the request header to meet a grayscale publish rule of `foo =^ bar $`, only requests with the request header can access the new-nginx service.



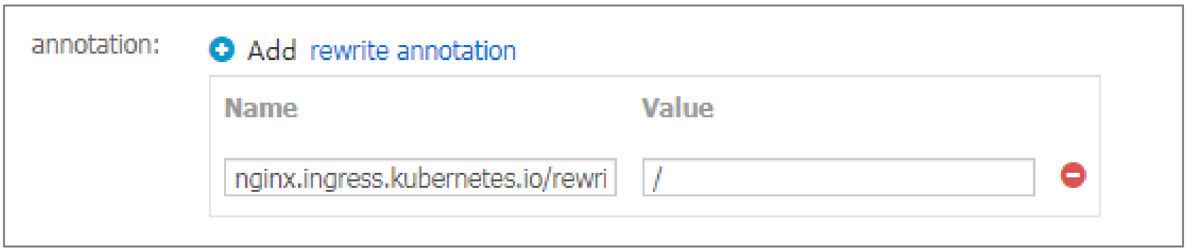
- Service: Routing rule configuration service.
- Type: matching request header, cookie, and query (request) parameters are supported.
- Name and match value: User-defined request field, name and match value are key-value pairs.
- Match rules: Regular and exact matches are supported.

7. Configure the annotations.

Click rewrite annotation, a typical redirection annotation can be added to the route. `nginx . ingress . kubernetes . io / rewrite - target : /` indicates that the / *path* is redirected to the root path / that the backend service can recognize.

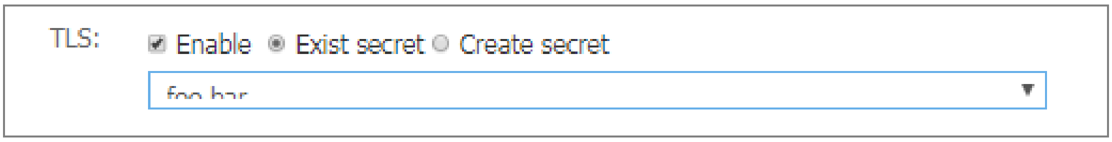
> **Note:**
> In this example, the access path is not configured, so no need to configure rewrite annotations. The purpose of the rewrite annotation is to enable Ingress to forward to the backend as the root path, avoiding 404 errors caused by incorrect access path configuration.

You can also click Add to enter the annotation name and value, which is the annotation key-value pair for Ingress. For more information, see https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/.



8. Configure TLS. Select Enable and configure the secure Ingress service. For more information, see Configure a safe routing service.

- You can select to use an existing secret.



a. Log on to the master node and create `tls . key` and `tls . crt` .

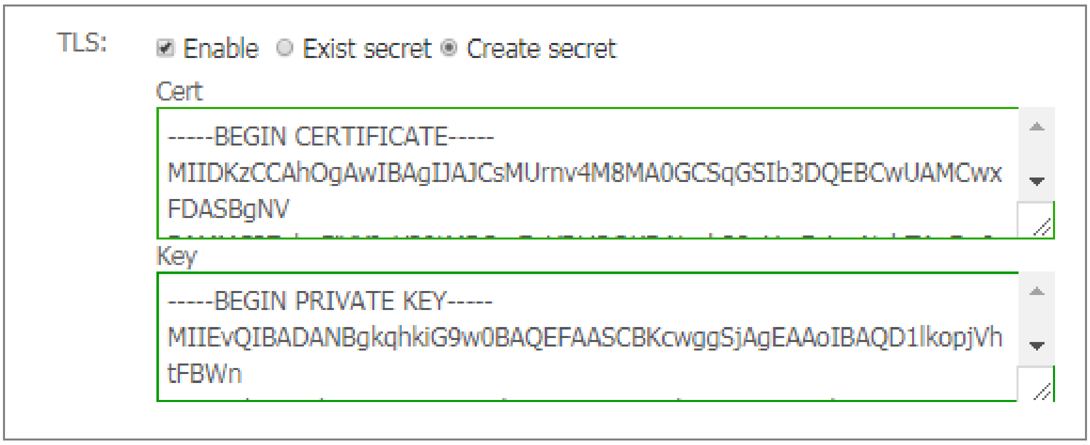
```
openssl req - x509 - nodes - days 365 - newkey rsa :
2048 - keyout tls . key - out tls . crt - subj "/ CN =
foo . bar . com / O = foo . bar . com "
```

b. Create a secret.

```
kubectl create secret tls foo . bar -- key tls .
key -- cert tls . crt
```

c. Run the `kubectl get secret` command to see that secret has been successfully created. You can use the secret that you have created in the Web interface, `foo . bar` .

- You can create the secret with one click by using the created TLS private key and certificate.



a. Log on to the master node and create `tls . key` and `tls . crt` .

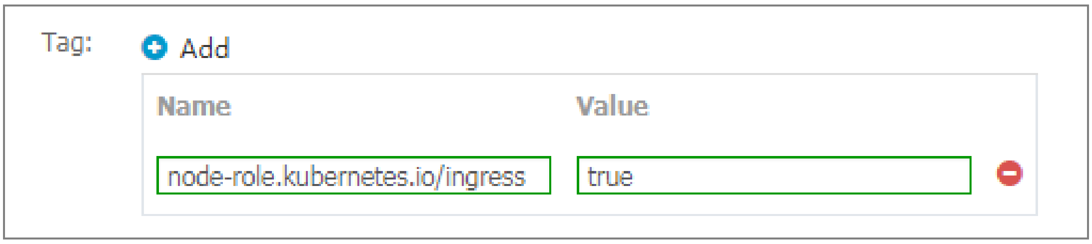
```
openssl req - x509 - nodes - days 365 - newkey rsa :
2048 - keyout tls . key - out tls . crt - subj "/ CN =
foo . bar . com / O = foo . bar . com "
```

b. Run the `vim tls . key` and `vim tls . crt` to get the generated private key and certificate.

c. Copy the generated certificate and private key to the Cert and Key fields.

9. Add the tags.

   Add the corresponding tags for Ingress to indicate the characteristics of the Ingress.



10. Click Create.

    The Ingress nginx-ingress is displayed on the Ingress page.

11. Click on the access domain name `test.[cluster-id].[region-id].alicontainer.com` in the route, and `foo.bar.com` to access the welcome page of nginx.

    Click on the route address pointing the new-nginx service and find the page that points the old-nginx application.

    Note:

Access the route address in the browser. By default, the request header does not have the `foo =^ bar $`, so the traffic is directed to the old-nginx application.

12. Log on to the master node by using SSH. Run the following command to simulate the access result with a specific request header.

```
curl  - H  " Host :  foo . bar . com "  http :// 47 . 107 . 20 . 35
old
 curl  - H  " Host :  foo . bar . com "  http :// 47 . 107 . 20 . 35
old
 curl  - H  " Host :  foo . bar . com "  http :// 47 . 107 . 20 . 35  #  Similar  to  browser  access  requests
old
 curl  - H  " Host :  foo . bar . com " - H  " foo :  bar "  http :// 47 . 107 . 20 . 35  #  Simulate  an  access  request  with  a  unique  header ,  returning  results  based  on  routing  weight
new
 curl  - H  " Host :  foo . bar . com " - H  " foo :  bar "  http :// 47 . 107 . 20 . 35
old
 curl  - H  " Host :  foo . bar . com " - H  " foo :  bar "  http :// 47 . 107 . 20 . 35
old
 curl  - H  " Host :  foo . bar . com " - H  " foo :  bar "  http :// 47 . 107 . 20 . 35
new
```

6.7 View Ingress details

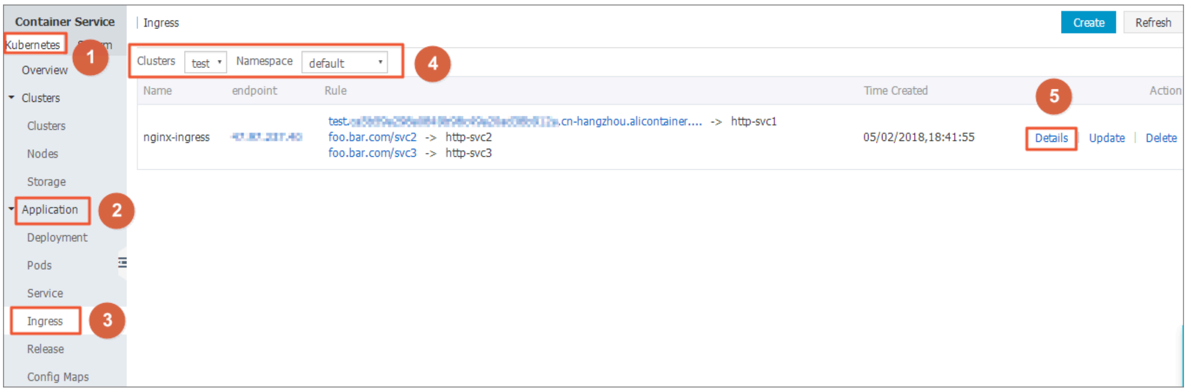
Prerequisites

- You have successfully created a Kubernetes cluster and Ingress controller is running normally in the cluster. For how to create a Kubernetes cluster, see Create a Kubernetes cluster.
- You have successfully created an Ingress. For more information, see Create an Ingress in the Container Service console.

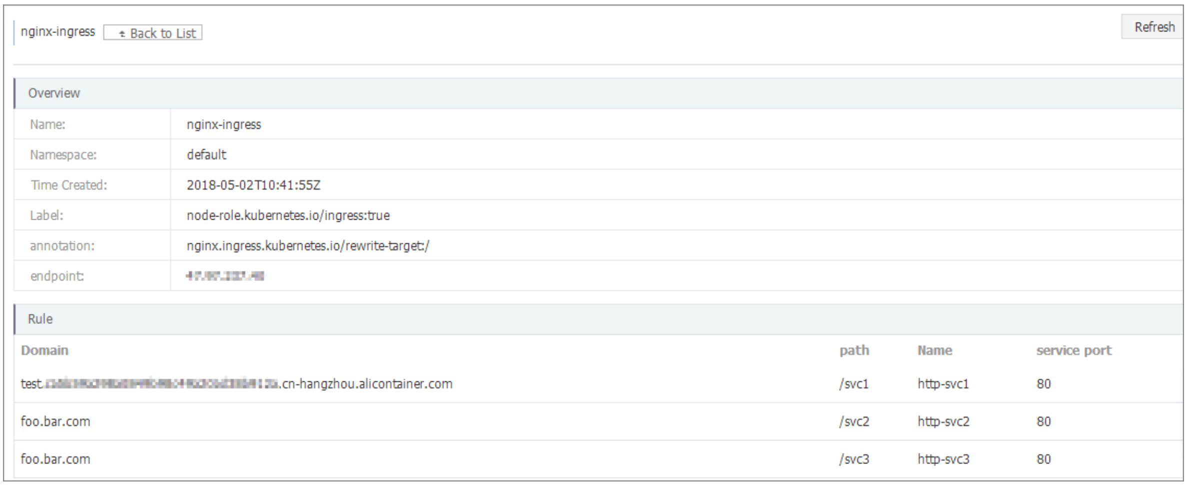
Procedure

1. Log on to the Container Service console.
2. Click Kubernetes Application > Ingress in the left-side navigation pane.

3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click Details at the right of the Ingress.



On the details page, you can view the overview and rules of the Ingress.



6.8 Update an Ingress

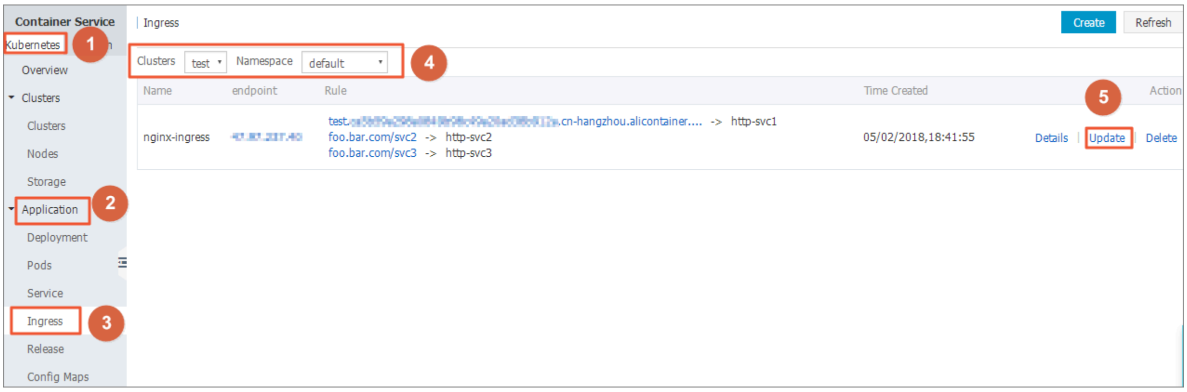
Prerequisites

- You have successfully created a Kubernetes cluster and Ingress controller is running normally in the cluster. For how to create a Kubernetes cluster, see Create a Kubernetes cluster.
- You have successfully created an Ingress. For more information, see Create an Ingress in the Container Service console.

Procedure

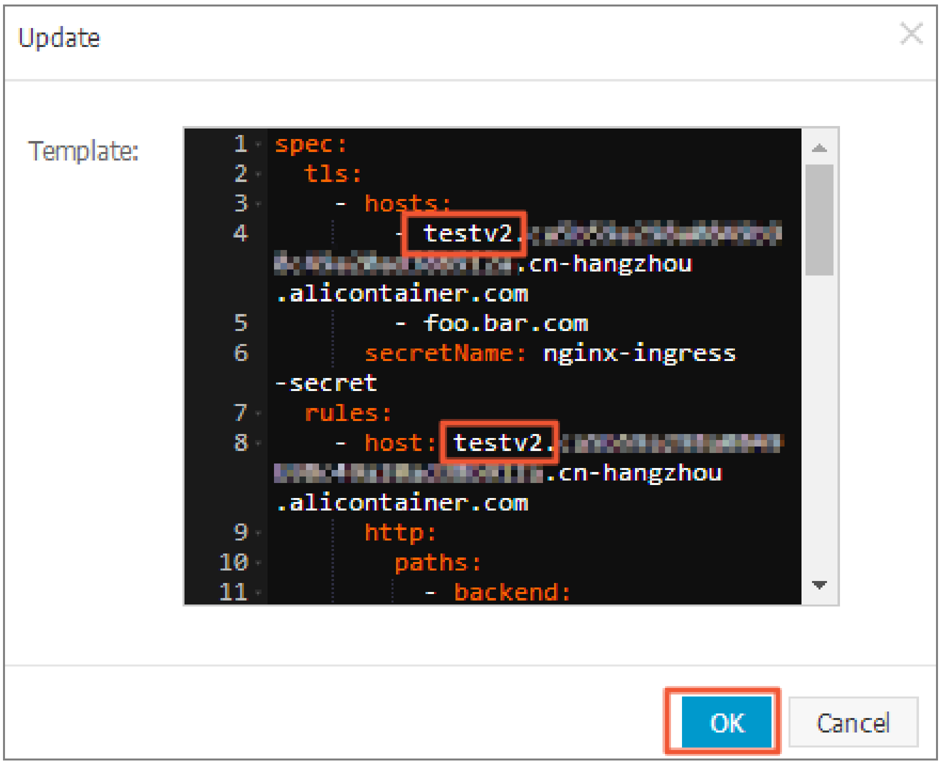
1. Log on to the Container Service console.
2. Under Kubernetes, click Application > Ingress in the left-side navigation pane.

3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click Update at the right of the Ingress.



4. Update the Ingress parameters in the displayed dialog box and then click OK.

change `test .[ cluster - id ].[ region - id ]. alicontain er . com` to `testv2 .[ cluster - id ].[ region - id ]. alicontain er . com`。



What's next

On the Ingress page, you can see a rule of this Ingress is changed.

6.9 Delete an Ingress

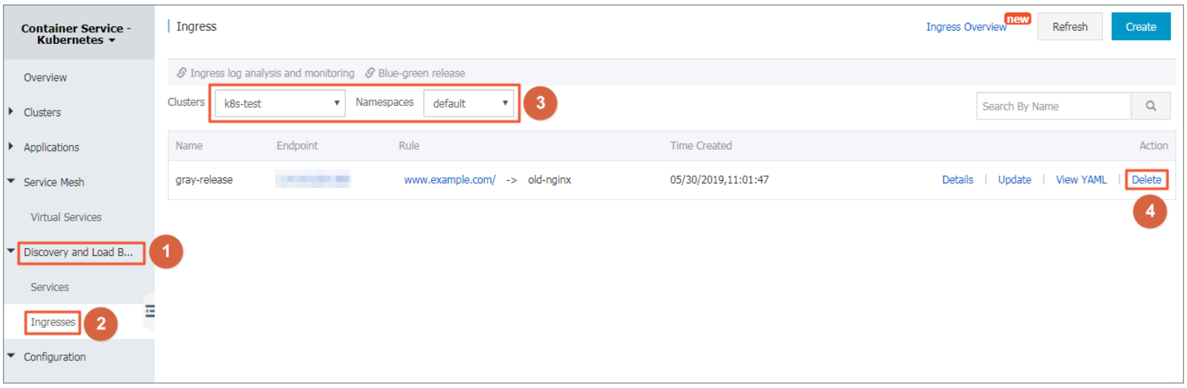
This topic describes how to delete an Ingress.

Prerequisites

- You have created a Kubernetes cluster and Ingress controller is running normally in the cluster. For more information, see Create a Kubernetes cluster.
- You have created an Ingress. For more information, see Create an Ingress in the Container Service console.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Discovery and Load Balancing > Ingresses.
3. Select the target cluster and namespace. Find the target Ingress, and then click Delete in the Action column.



4. In the displayed dialog box, click Confirm.

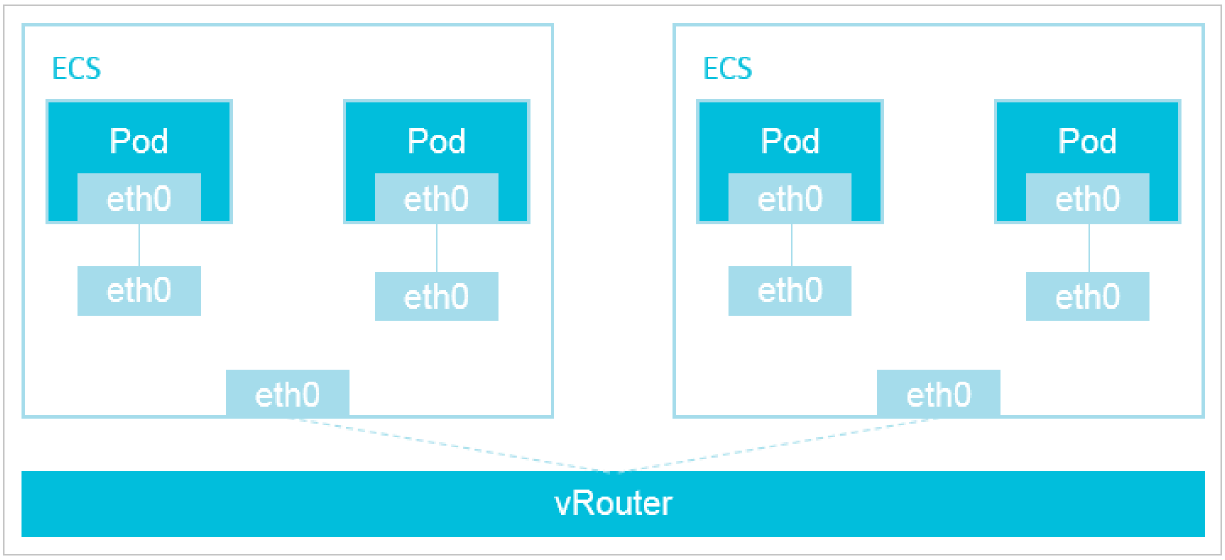
6.10 Terway network plugin

This topic describes how to use the Terway network plugin in a Kubernetes cluster that runs on Alibaba Cloud Container Service.

Terway network plugin

Terway, a network plugin developed by Alibaba Cloud Container Service, is fully compatible with Flannel, and provides the following features:

- Allocates Alibaba Cloud Elastic Network Interfaces (ENIs) to containers.
- Defines the access policies for containers according to the Kubernetes Network Policy. This network plugin is also compatible with the Calico Network Policy.

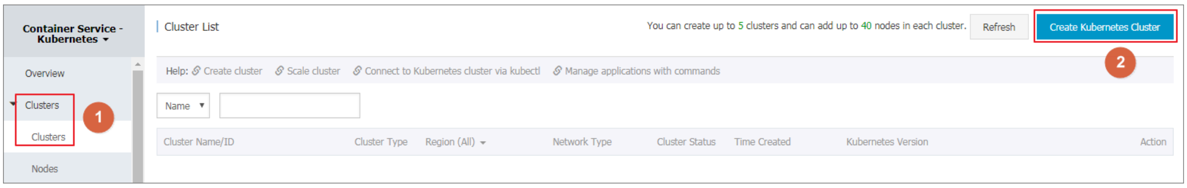


If you install the Terway network plugin in a Kubernetes cluster, each pod then has its own network stack and an IP address. Packets between pods on one ECS instance are forwarded directly by the instance. Packets between pods on different ECS instances are forwarded through the VRouter of a VPC. The Terway network plugin delivers high communication performance because it does not use tunneling technologies such as VXLAN to encapsulate packets.

Use the Terway network plugin

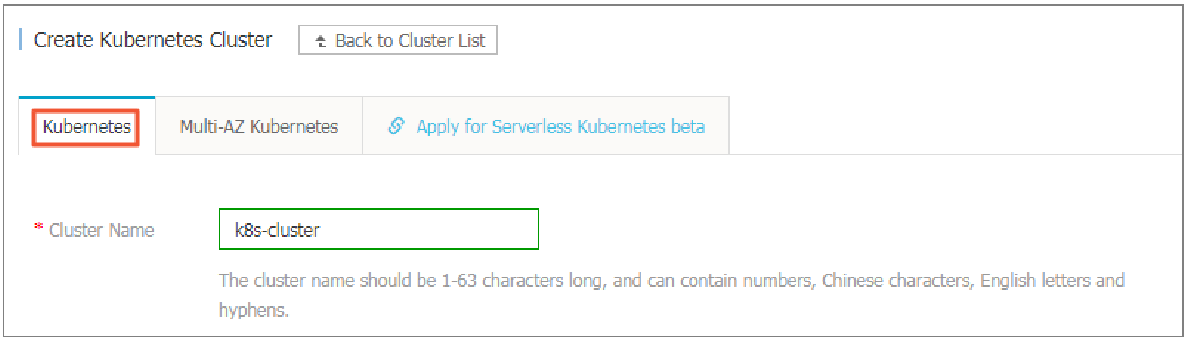
1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, click Clusters.

3. In the upper-right corner, click Create Kubernetes Cluster.

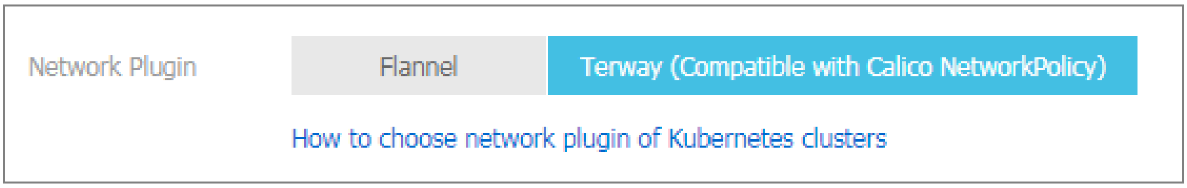


By default, the Create Kubernetes Cluster page is displayed.

Note:
In this example, a dedicated Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.



4. Select the Terway network plugin.



Flannel and Terway

Alibaba Cloud Container Service for Kubernetes provides two types of network plugins for you to create a Kubernetes cluster: Terway and Flannel.

Note:
For how to select a network plugin, see Do I select the Terway or Flannel plugin for my Kubernetes cluster network?

- Flannel: a simple and stable community Flannel CNI plugin. Flannel can interoperate with the high-speed network of Alibaba Cloud VPC to provide a high-performance and stable container network for clusters. However, it provides a limited amount of features. For example, it does not support the Kubernetes Network Policy.

- Terway: a network plugin developed by Alibaba Cloud Container service. It is fully compatible with Flannel, and can allocate Alibaba Cloud Elastic Network Interfaces (ENIs) to containers. It can also define the access policies between containers according to the Kubernetes Network Policy. In addition, you can use this network plugin to limit the bandwidth traffic of a single container. If you do not need to use the Network Policy, we recommend that you select Flannel. In other cases, we recommend that you select Terway.

  Note:

  - Terway provides the same Network Policy as Calico because Terway is integrated with the Felix component of Calico. If you create a cluster to use Calico, you can use Terway to switch to Alibaba Cloud Container Service for Kubernetes.
  - Terway is integrated with the Felix component V2.6.6.

6.11 Associate an ENI with a pod

This topic describes how to associate an Elastic Network Interface (ENI) with a pod.

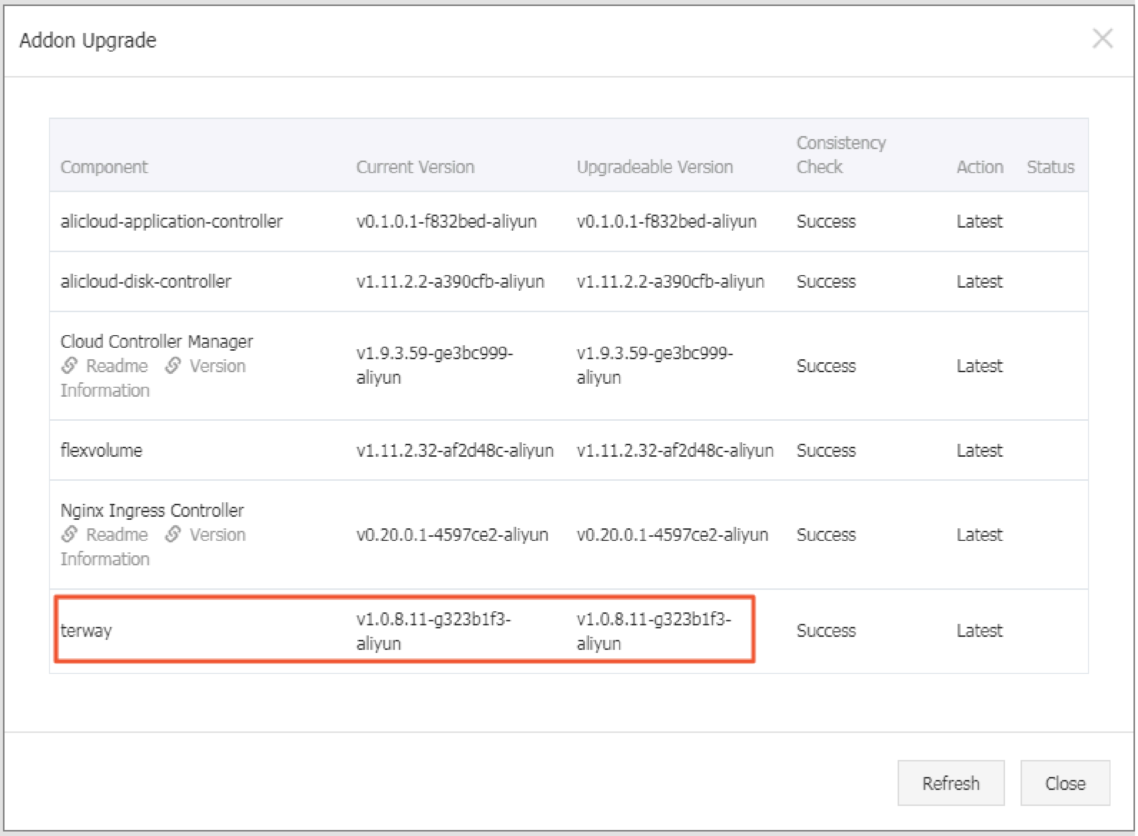
Context

- When you create a Kubernetes cluster, you need to select Network Plugin as Terway. For more information, see Create a Kubernetes cluster.
- If you use a Kubernetes cluster that is installed with the Terway network plugin, you must make sure that the Terway plugin is V1.0.0.1 or later.

  Note:

  1. Log on to the Container Service console, click Clusters under the Kubernetes menu.
  2. In the action column of the target cluster, choose More > Addon Upgrade.
  3. On the Addon Upgrade page, view your current version of Terway.

4. Determine whether to upgrade according to Current Version and Upgradeable Version. If you want to upgrade Terway, click Upgrade in the action column.



Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Deployment.
3. In the upper-right corner, click Create by Template.

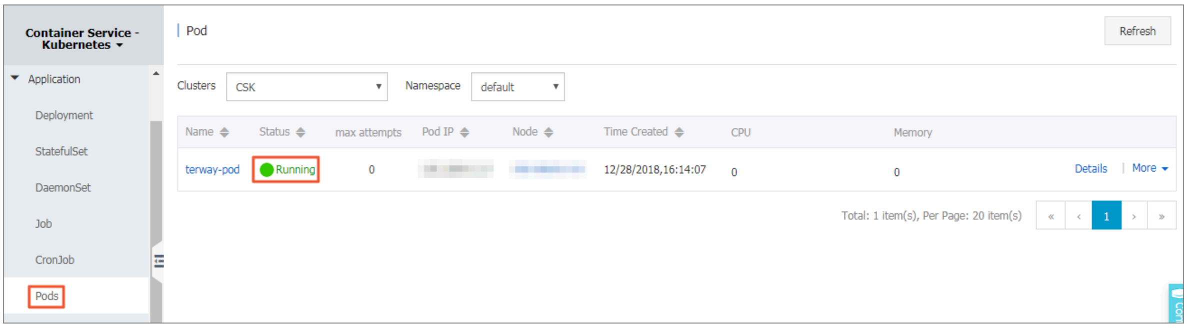
   You can use the following YAML template to create a pod:

```
apiVersion :  v1
kind :  Pod
metadata :
  name :  terway - pod
  labels :
    app :  nginx
spec :
  containers :
 -  name :  nginx
    image :  nginx
    ports :
   -  containerP  ort :  80
    resources :
      limits :
        aliyun / eni :  1
```

Result

1. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Pods.

   The pod named terway-pod is displayed.



2. In the left-side navigation pane under Kubernetes, click Clusters.
3. Click the name of the target cluster to view the cluster details.
4. In the Cluster Resource area, click VPC to view the VPC CIDR block of the cluster.
5. Run the following command to obtain the IP address of the deployed pod and verify that the IP address is within the VPC CIDR block of the cluster:

```
$  kubectl   get   pod  - o   wide
```

6.12 Use a network policy

Prerequisites

- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have selected the Terwaynetwork plugin when creating the Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have connected to the Kubernetes cluster by using kubectl, see Connect to a Kubernetes cluster by using kubectl.

Verify that an Nginx service is accessible to pods

1. Run the following command to create an Nginx application and expose it through a service named Nginx:

```
$  kubectl   run   nginx  -- image = nginx
 deployment . apps / nginx   created
$  kubectl   get   pod
 NAME                       READY     STATUS      RESTARTS     AGE
 nginx - 64f497f8fd - znbxb      1 / 1        Running      0
 45s
```

```
$ kubectl expose deployment nginx -- port = 80
 service / nginx exposed
$ kubectl get service
 NAME           TYPE          CLUSTER - IP      EXTERNAL - IP
 PORT ( S )    AGE
 kubernetes     ClusterIP     172 . 19 . 0 . 1     < none >
 443 / TCP     3h
 nginx          ClusterIP     172 . 19 . 8 . 48    < none >
 80 / TCP      10s
```

2. Run the following command to create a pod named busybox and use the pod to access the Nginx service created in step 1:

```
$ kubectl run busybox -- rm - ti -- image = busybox / bin /
 sh
 kubectl run -- generator = deployment / apps . v1beta1 is
 DEPRECATED and will be removed in a future version
 . Use kubectl create instead .
 If you don ' t see a command prompt , try pressing
 enter .
/ # wget nginx
 Connecting to nginx ( 172 . 19 . 8 . 48 : 80 )
 index . html            100 % |
***************************************************************************
    612   0 : 00 : 00   ETA
/ #
```

Use a network policy to set the Nginx service to be accessible only to a specifically labeled application

1. Run the following command to create a *policy . yaml* file:

```
$ vim policy . yaml
 kind : NetworkPol icy
 apiVersion : networking . k8s . io / v1
 metadata :
   name : access - nginx
 spec :
   podSelecto r :
     matchLabel s :
       run : nginx
   ingress :
  - from :
    - podSelecto r :
        matchLabel s :
          access : " true "
```

2. Run the following command to create a network policy according to the *policy . yaml* file created in step 1:

```
$ kubectl apply - f policy . yaml
```

```
networkpol  icy . networking . k8s . io / access - nginx   created
```

3. Run the following command to verify that the Nginx service cannot be accessed if you do not define any access label in the command:

```
$  kubectl   run   busybox  -- rm  - ti  -- image = busybox  / bin /
 sh
 If   you   don ' t   see   a   command   prompt ,  try   pressing
 enter .
/ #  wget   nginx
 Connecting   to   nginx  ( 172 . 19 . 8 . 48 : 80 )
 wget :  can ' t   connect   to   remote   host  ( 172 . 19 . 8 . 48
 ):  Connection   timed   out
/ #
```

4. Run the following command to verify that the Nginx service can be accessed if an access label is defined in the command:

```
$  kubectl   run   busybox  -- rm  - ti  -- labels =" access = true
 " -- image = busybox  / bin / sh
 If   you   don ' t   see   a   command   prompt ,  try   pressing
 enter .
/ #  wget   nginx
 Connecting   to   nginx  ( 172 . 19 . 8 . 48 : 80 )
 index . html             100 % |
************************************************************************
    612    0 : 00 : 00   ETA
/ #
```

Use a network policy to specify a source IP CIDR block that can access a service exposed by an SLB service over the Internet

1. Run the following command to create an Alibaba Cloud SLB service for the preceding Nginx application, that is, specify `type = LoadBalanc  er` to expose the Nginx service to the Internet:

```
$  vim   nginx - service . yaml
 apiVersion :  v1
 kind :  Service
 metadata :
   labels :
     run :  nginx
   name :  nginx - slb
 spec :
   externalTr  afficPolic  y :  Local
   ports :
  -  port :  80
     protocol :  TCP
     targetPort :  80
   selector :
     run :  nginx
   type :  LoadBalanc  er

$  kubectl   apply   - f   nginx - service . yaml
 service / nginx - slb   created

$  kubectl   get   service   nginx - slb
```

```
NAME           TYPE             CLUSTER - IP        EXTERNAL - IP
    PORT ( S )          AGE
nginx - slb     LoadBalanc  er     172 . 19 . 12 . 254     47 . 110
. 200 . 119     80 : 32240 / TCP     8m
```

2. Run the following command to verify that the IP address of the created SLB service, that is, 47.110.200.119, cannot be accessed:

```
$  wget   47 . 110 . 200 . 119
-- 2018 - 11 - 21   11 : 46 : 05 --   http :// 47 . 110 . 200 . 119
 /
 Connecting   to   47 . 110 . 200 . 119 : 80 ...  failed :
 Connection   refused .
```

Note:

Access failure occurs due to the following reasons:

- You have configured access to the Nginx service only for the applications labeled with `access = true`.
- You have attempted to access the IP address of the SLB instance from outside the Kubernetes system. This is different from Use a network policy to set the Nginx service to be accessible only to a specifically labeled application.

Solution: Modify the network policy and add a source IP CIDR block that is allowed to access the Nginx service.

3. Run the following command to view your local IP address:

```
$  curl   myip . ipip . net

 IP   address :  10 . 0 . 0 . 1   from :  China   Beijing   Beijing
       # The   local   IP   address   varies   by   devices .
```

4. Run the following command to modify the created *policy . yaml* file:

```
$  vim   policy . yaml
 kind :  NetworkPol  icy
 apiVersion :  networking . k8s . io / v1
 metadata :
   name :  access - nginx
 spec :
   podSelecto  r :
     matchLabel  s :
       run :  nginx
   ingress :
  -  from :
    -  podSelecto  r :
         matchLabel  s :
           access : " true "
    -  ipBlock :
         cidr :  100 . 64 . 0 . 0 / 10
    -  ipBlock :
```

```
        cidr : 10 . 0 . 0 . 1 / 24        # Set   the   CIDR
 block   to   which   the   local   IP   address   belongs .  This
 is   an   example .  Set   the   required   parameters   according
   to   your   device .

$  kubectl   apply  - f   policy . yaml
 networkpol  icy . networking . k8s . io / access - nginx
 unchanged
```

Note:

- The outgoing interface of a network may have multiple IP addresses. We recommend that you specify an entire CIDR block.
- The SLB health check address belongs to the `100 . 64 . 0 . 0 / 10` CIDR block. Therefore, you must specify the `100 . 64 . 0 . 0 / 10` CIDR block.

5. Run the following command to verify that the Nginx service can be accessed:

```
$   kubectl   run   busybox  -- rm  - ti  -- labels =" access = true
 " -- image = busybox  / bin / sh

 If   you   don ' t   see   a   command   prompt ,  try   pressing
 enter .
/ #  wget   47 . 110 . 200 . 119
 Connecting   to   47 . 110 . 200 . 119  ( 47 . 110 . 200 . 119 : 80
 )
 index . html             100 % |
***********************************************************|    612
   0 : 00 : 00   ETA
/ #
```

Use a network policy to set a pod that can access only *www . aliyun . com*

1. Run the following command to obtain the IP address list resolved from the domain name of *www . aliyun . com* :

```
$  dig  + short   www . aliyun . com
 www - jp - de - intl - adns . aliyun . com .
 www - jp - de - intl - adns . aliyun . com . gds . alibabadns . com
 .
 v6wagbridg  e . aliyun . com .
 v6wagbridg  e . aliyun . com . gds . alibabadns . com .
 106 . 11 . 93 . 21
 140 . 205 . 32 . 4
 140 . 205 . 230 . 13
 140 . 205 . 34 . 3
```

2. Run the following command to create a *busybox - policy* file:

```
$  vim   busybox - policy . yaml
 kind :  NetworkPol  icy
 apiVersion :  networking . k8s . io / v1
 metadata :
   name :  busybox - policy
 spec :
   podSelecto  r :
```

```
    matchLabel  s :
      run :  busybox
  egress :
 -  to :
   -  ipBlock :
        cidr :  106 . 11 . 93 . 21 / 32
   -  ipBlock :
        cidr :  140 . 205 . 32 . 4 / 32
   -  ipBlock :
        cidr :  140 . 205 . 230 . 13 / 32
   -  ipBlock :
        cidr :  140 . 205 . 34 . 3 / 32
 -  to :
   -  ipBlock :
        cidr :  0 . 0 . 0 . 0 / 0
    ports :
   -  protocol :  UDP
      port :  53
```

Note:

In the preceding *busybox - policy* **file, an egress rule is set to specify the CIDR blocks that can be accessed by cluster applications. You need to set the condition that UDP requests are allowed. Otherwise, DNS resolution will fail.**

3. **Run the following command to create a network policy according to the** *busybox - policy* **file:**

```
$  kubectl   apply  - f   busybox - policy . yaml
 networkpol  icy . networking . k8s . io / busybox - policy
 created
```

4. **Run the following command to verify that no website (for example,** *www . google . com* **) can be accessed except for** *www . aliyun . com* **:**

```
$  kubectl   run   busybox   -- rm   - ti  -- image = busybox   / bin /
 sh
 If   you   don ' t   see   a   command   prompt ,  try   pressing
 enter .
/ #  wget   www . google . com
 Connecting   to   www . google . com  ( 64 . 13 . 192 . 74 : 80 )
 wget :  can ' t   connect   to   remote   host  ( 64 . 13 . 192 .
 74 ):  Connection   timed   out
```

5. **Run the following command to verify that** *www . aliyun . com* **can be accessed:**

```
/ #  wget   www . aliyun . com
 Connecting   to   www . aliyun . com  ( 140 . 205 . 34 . 3 : 80 )
 Connecting   to   www . aliyun . com  ( 140 . 205 . 34 . 3 : 443 )
 wget :  note :  TLS   certificat  e   validation   not
 implemente  d
 index . html              100 % |
************************************************************|   462k
   0 : 00 : 00   ETA
/ #
```

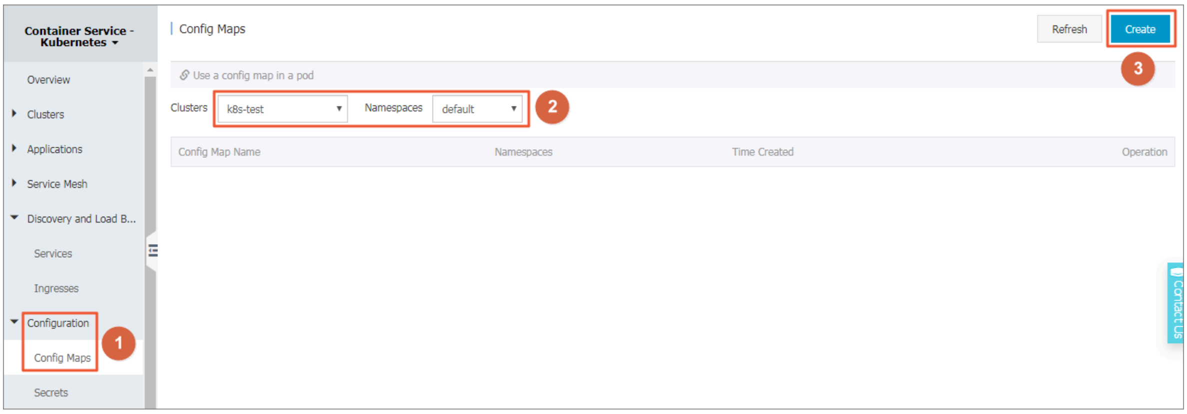
7 Config Map and Secret management

7.1 Create a Config Map

In the Container Service console, you can create a Config Map on the Config Maps page or by using a template.

Create a Config Map on Config Maps page

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Config Maps.
3. Select the target cluster and namespace. Then, in the upper-right corner, click Create.

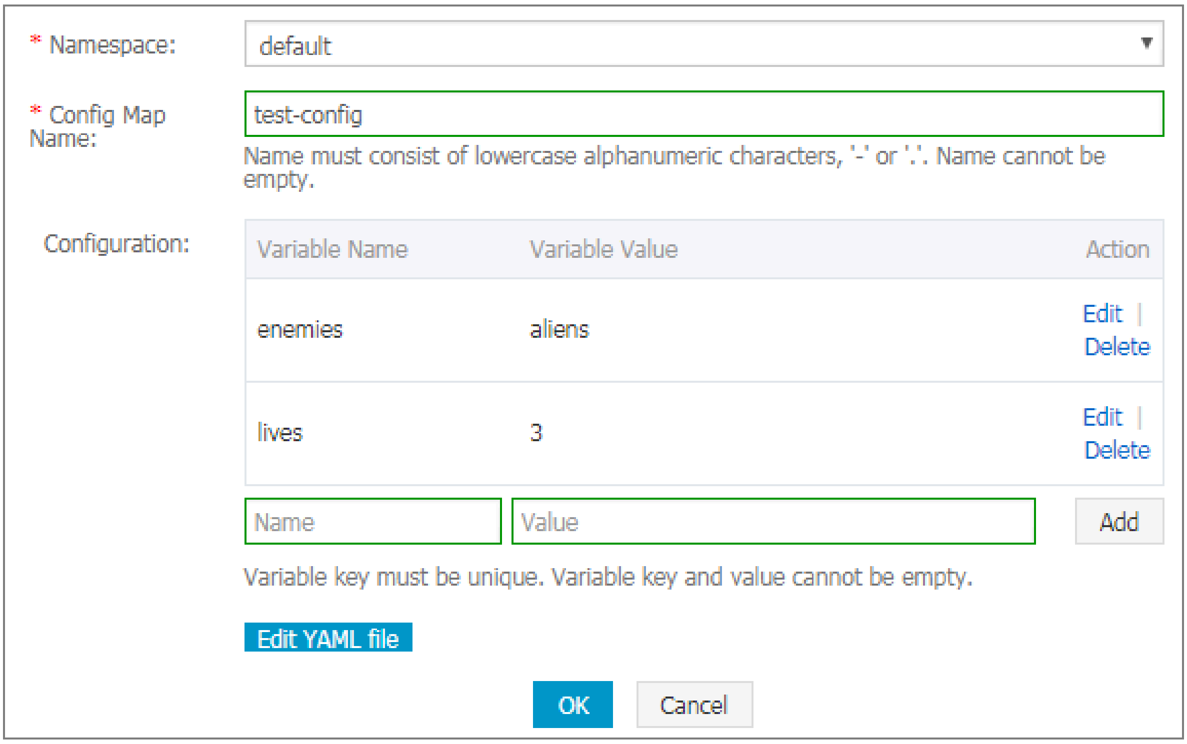


4. Complete the settings and then click OK.

    - Namespace: Select the namespace to which the Config Map belongs. Config Map is a Kubernetes resource object that must be applied to the namespace.
    - Config Map Name: Enter the Config Map name, which can contain lowercase letters, numbers, hyphens (-), and periods (.). The name cannot be empty.

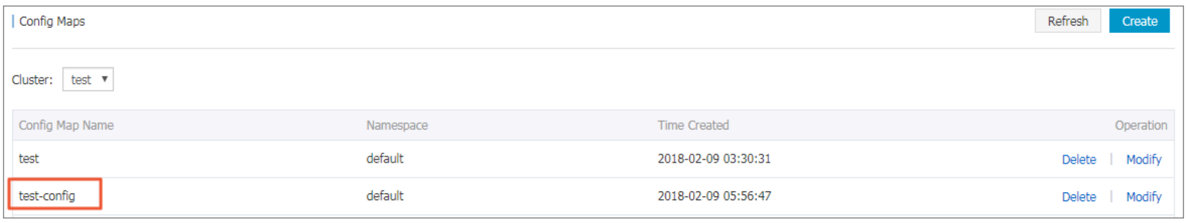
Other resource objects must reference the Config Map name to obtain the configuration information.

- Configuration: Enter the Variable Name and the Variable Value. Then, click Add on the right. You can also click Edit, complete the configuration in the displayed dialog box, and click OK.



In this example, configure the variables enemies and lives to pass the parameters aliens and 3 respectively.

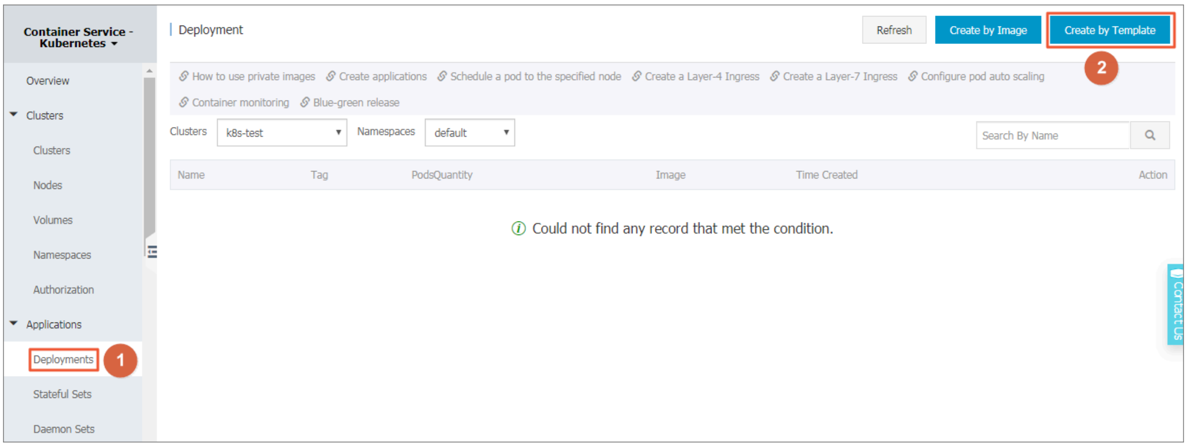
5. You can view the Config Map test-config on the Config Maps page after clicking OK.



Create a Config Map by using a template

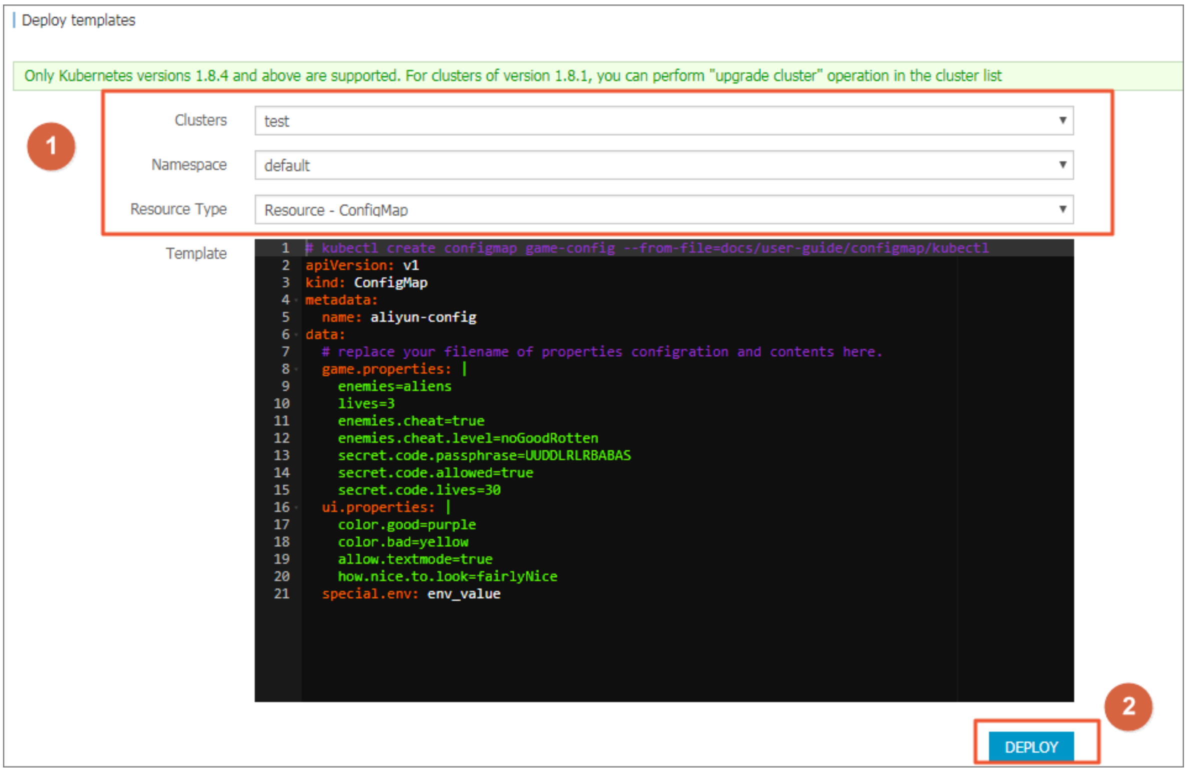
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments.

3. Click Create by template in the upper-right corner.

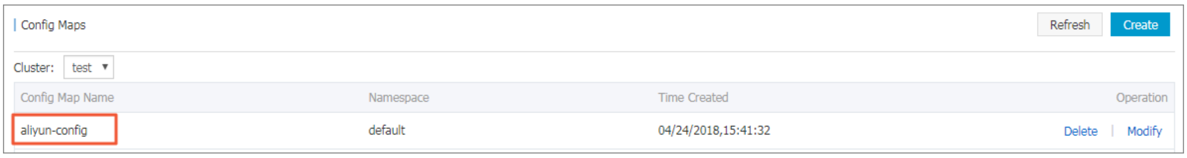


4. On the Deploy templates page, complete the settings and then click DEPLOY.

   - Clusters: Select the cluster in which the Config Map is to be created.
   - Namespace: Select the namespace to which the Config Map belongs. Config map is a Kubernetes resource object that must be applied to the namespace.
   - Resource Type: You can write your own Config Map based on the Kubernetes YAML syntax rules, or select the sample template resource-ConfigMap. In the sample template, the Config Map is named as aliyun-config and includes two variable files `game . properties` and `ui . properties`. You can make modifications based on the sample template. Then, click DEPLOY.



5. After the deployment, you can view the Config Map aliyun-config on the Config Maps page.



7.2 Use a config map in a pod

You can use a config map in a pod in the following scenarios:

- Use a config map to define the pod environment variables.
- Use a config map to configure command line parameters.
- Use a config map in data volumes.

For more information, see Configure a pod to use a ConfigMap.

Limits

To use a config map in a pod, make sure the config map and the pod are in the same cluster and namespace.

Create a config map

In this example, create a config map special-config, which includes two key-value pairs: `SPECIAL_LE VEL : very` and `SPECIAL_TY PE : charm`.

Create a config map by using an orchestration template

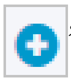
1. Log on to the Container Service console.
2. Under Kubernetes, click Application > Deployment. Click Create by template in the upper-right corner.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click DEPLOY.

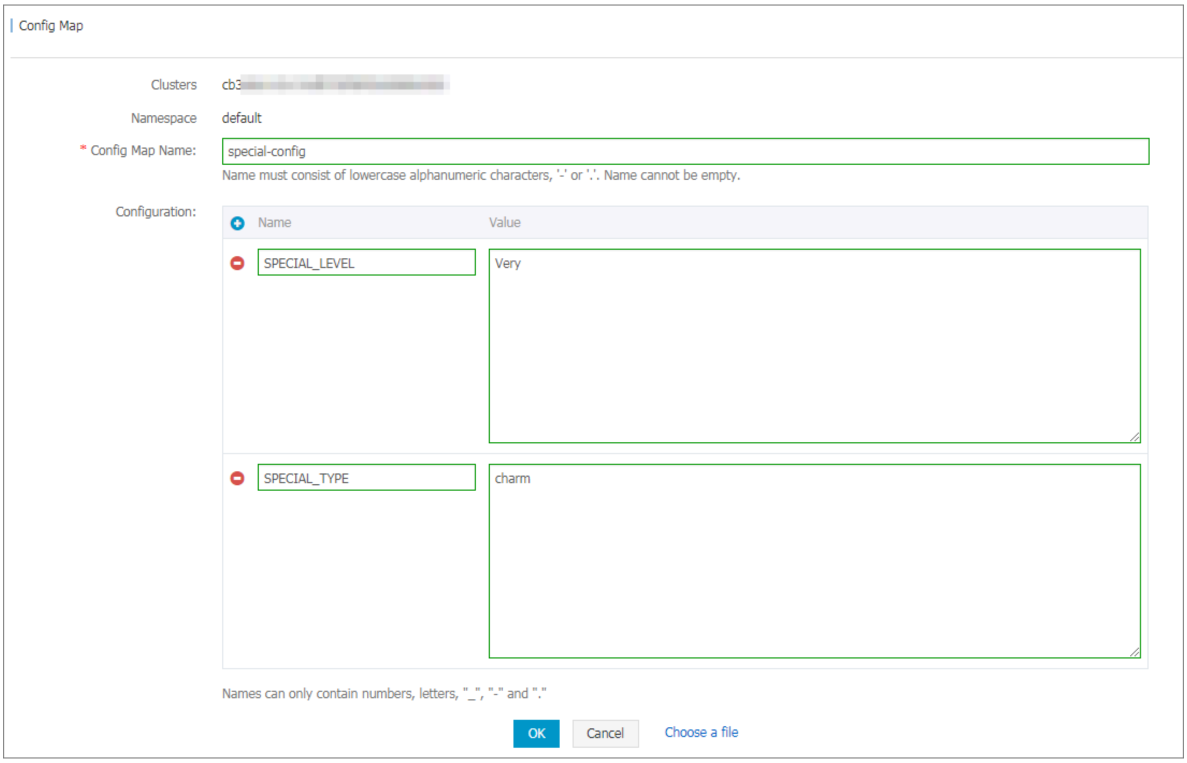
   You can use the following YAML sample template to create a config map.

```
apiVersion : v1
kind : ConfigMap
metadata :
   name : special - config
   namespace : default
data :
   SPECIAL_LE VEL : very
```

```
SPECIAL_TY PE : charm
```

Create a config map on Config Maps page

1. Log on to the Container Service console.
2. Under Kubernetes, choose Configuration > Config Maps in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click Create in the upper-right corner.
4. Enter a config map name, click the plus icon , set the name and value for each entry of the config map, and then click OK.



Use a config map to define pod environment variables

Use config map data to define pod environment variables

1. Log on to the Container Service console.
2. Under Kubernetes, clickApplication > Deployment . Click Create by template in the upper-right corner.

3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click DEPLOY.

   You can define the environment variables in a pod. Use `valueFrom` to reference the value of SPECIAL_LEVEL to define the pod environment variables.

   See the following orchestration 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_LE  VEL_KEY
           valueFrom :                             ## Use
valueFrom   to   specify   env   to   reference   the   value   of
  the   config   map .
             configMapK  eyRef :
               name :  special - config                ## The
referenced   config   map   name .
               key :  SPECIAL_LE  VEL                  ## The
referenced   config   map   key .
   restartPol  icy :  Never
```

Similarly, to define the values of multiple config maps to the environment variable values of the pod, add multiple env parameters in the pod.

Configure all key-value pairs of a config map to pod environment variables

1. Log on to the Container Service console.
2. Under Kubernetes, click Application > Deployment. Click Create by template in the upper-right corner.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click DEPLOY.

   To configure all the key-value pairs of a config map to the environment variables of a pod, use the envFrom parameter. The key in a config map becomes the environment variable name in the pod.

   See the following orchestration 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   the   key -
value   pairs   in   the   config   map   special - config .
      - configMapR  ef :
          name :  special - config
   restartPol  icy :  Never
```

Use a config map to configure command line parameters

1. Log on to the Container Service console.
2. Under Kubernetes, click Application > Deployment . Click Create by template in the upper-right corner.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click DEPLOY.

   You can use the config map to configure the commands or parameter values in the container by using the environment variable replacement syntax `$( VAR_NAME )`.

   See the following orchestration example:

```
apiVersion :  v1
kind :  Pod
metadata :
   name :  config - pod - 3
spec :
   containers :
    - name :  test - container
      image :  busybox
      command : [ "/ bin / sh ", "- c ", " echo  $( SPECIAL_LE
VEL_KEY ) $( SPECIAL_TY  PE_KEY )" ]
      env :
       - name :  SPECIAL_LE  VEL_KEY
         valueFrom :
           configMapK  eyRef :
             name :  special - config
             key :  SPECIAL_LE  VEL
       - name :  SPECIAL_TY  PE_KEY
         valueFrom :
           configMapK  eyRef :
             name :  special - config
             key :  SPECIAL_TY  PE
```

```
    restartPol  icy :  Never
```

The output after running the pod is as follows:

```
very  charm
```

Use a config map in data volumes

1. Log on to the Container Service console.
2. Under the Kubernetes menu, click Application Deployment in the left-side navigation pane. Click Create by template in the upper-right corner.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click DEPLOY.

   You can also use a config map in data volumes. Specifying the config map name under volumes stores the key-value pair data to the mountPath directory (*/ etc / config* in this example). It finally generates a configuration file with key as the file name and values as the contents of the file.

   Then, the configuration file with key as the name and value as the contents is generated.

```
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  file  names  under  this  directory .
       volumeMoun  ts :
       -  name :  config - volume
         mountPath : / etc / config
   volumes :
    -  name :  config - volume
       configMap :
         name :  special - config
   restartPol  icy :  Never
```

Keys of the config map are output after running the pod.

```
SPECIAL_TY  PE
```

```
SPECIAL_LE  VEL
```

7.3 View a ConfigMap

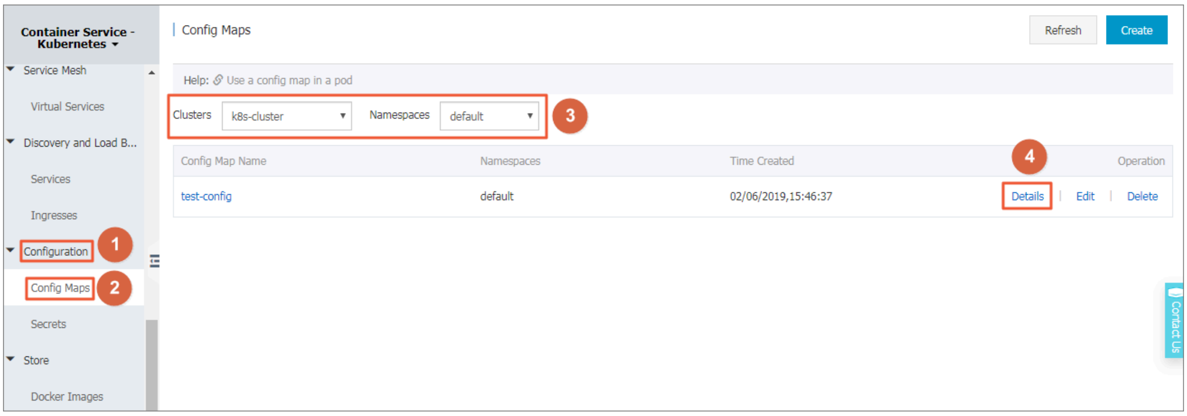
This topic describes how to view a created ConfigMap by using the Container Service console of Alibaba Cloud.

Prerequisites

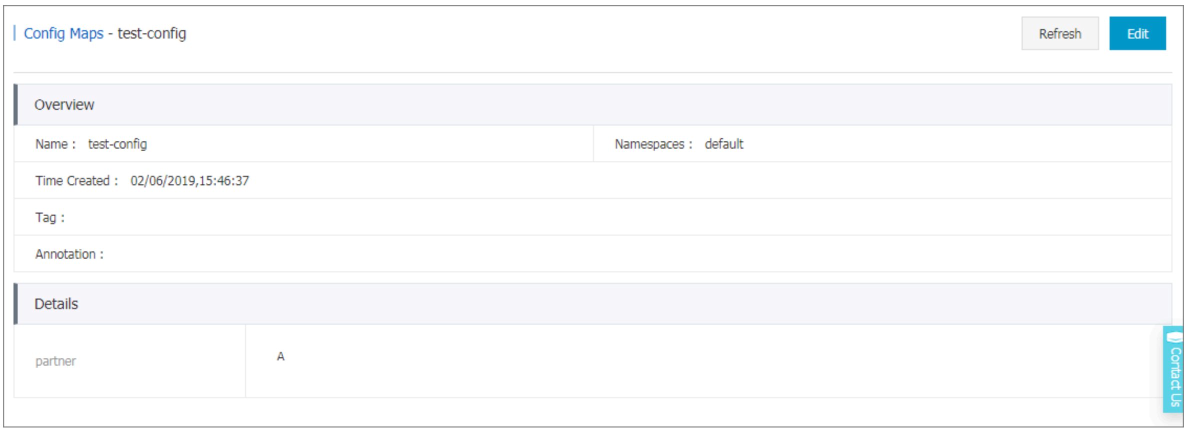
- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A ConfigMap is created. For more information, see Create a Config Map.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Config Maps.
3. Select the target cluster and namespace, find the target ConfigMap, and then click Details in the Operation column.



Then, you can view the details of the ConfigMap.



7.4 Modify a Config Map

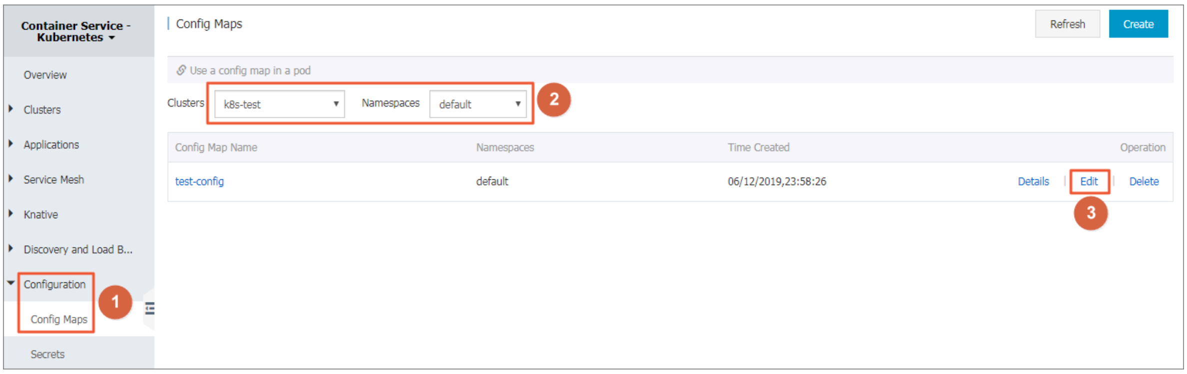
You can modify the configurations of a Config Map.

Note:

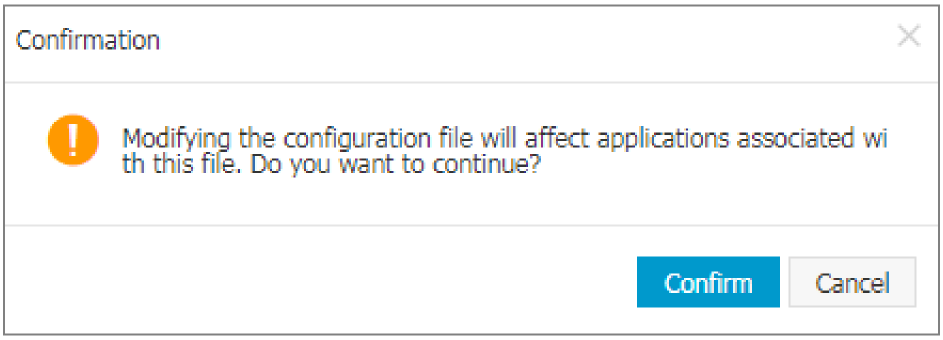
Modifying a Config Map affects applications that use this Config Map.

Modify a Config Map on the Config Maps page

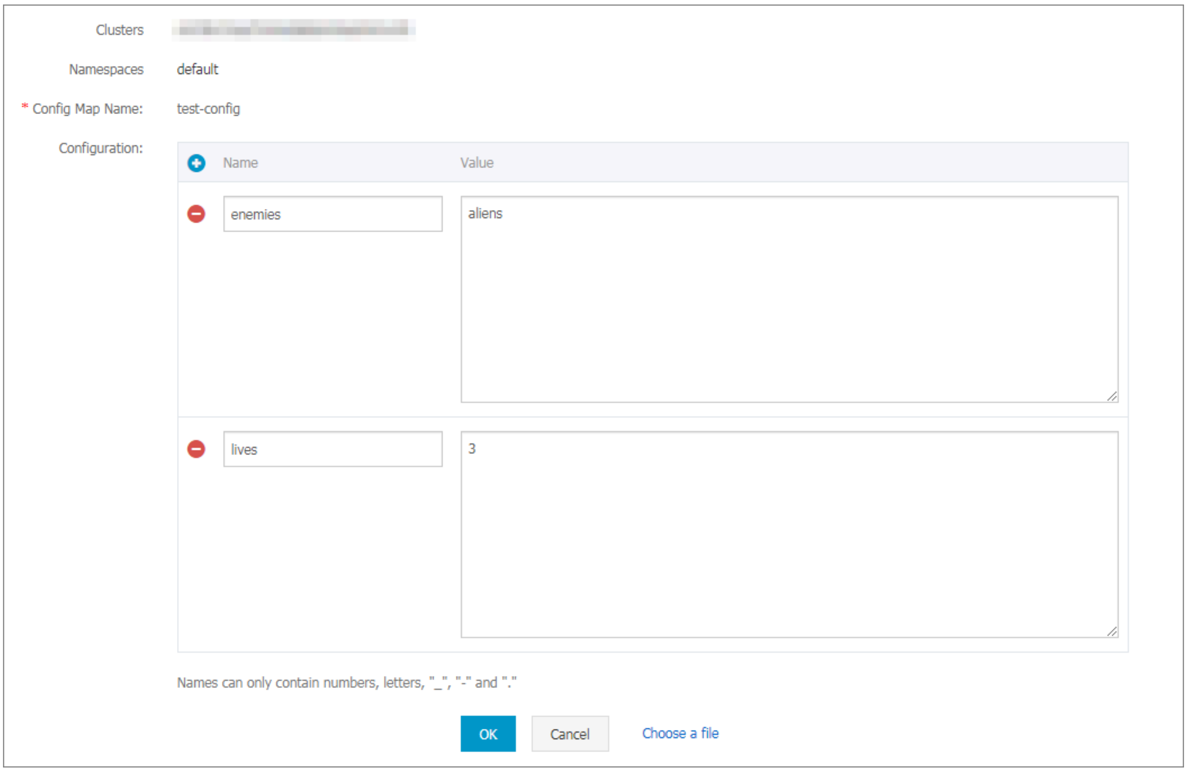
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Config Maps.
3. Select the target cluster and namespace, and find the target Config Map. Then, in the Operation column, click Edit.



4. In the displayed dialog box, click Confirm



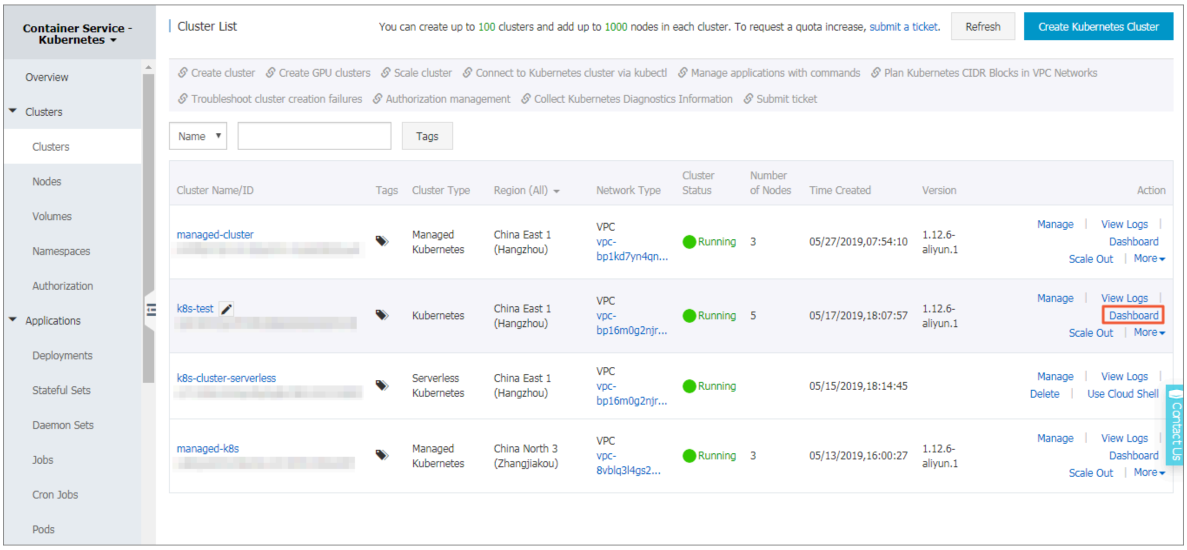
5. Modify the configurations.



6. Click OK.

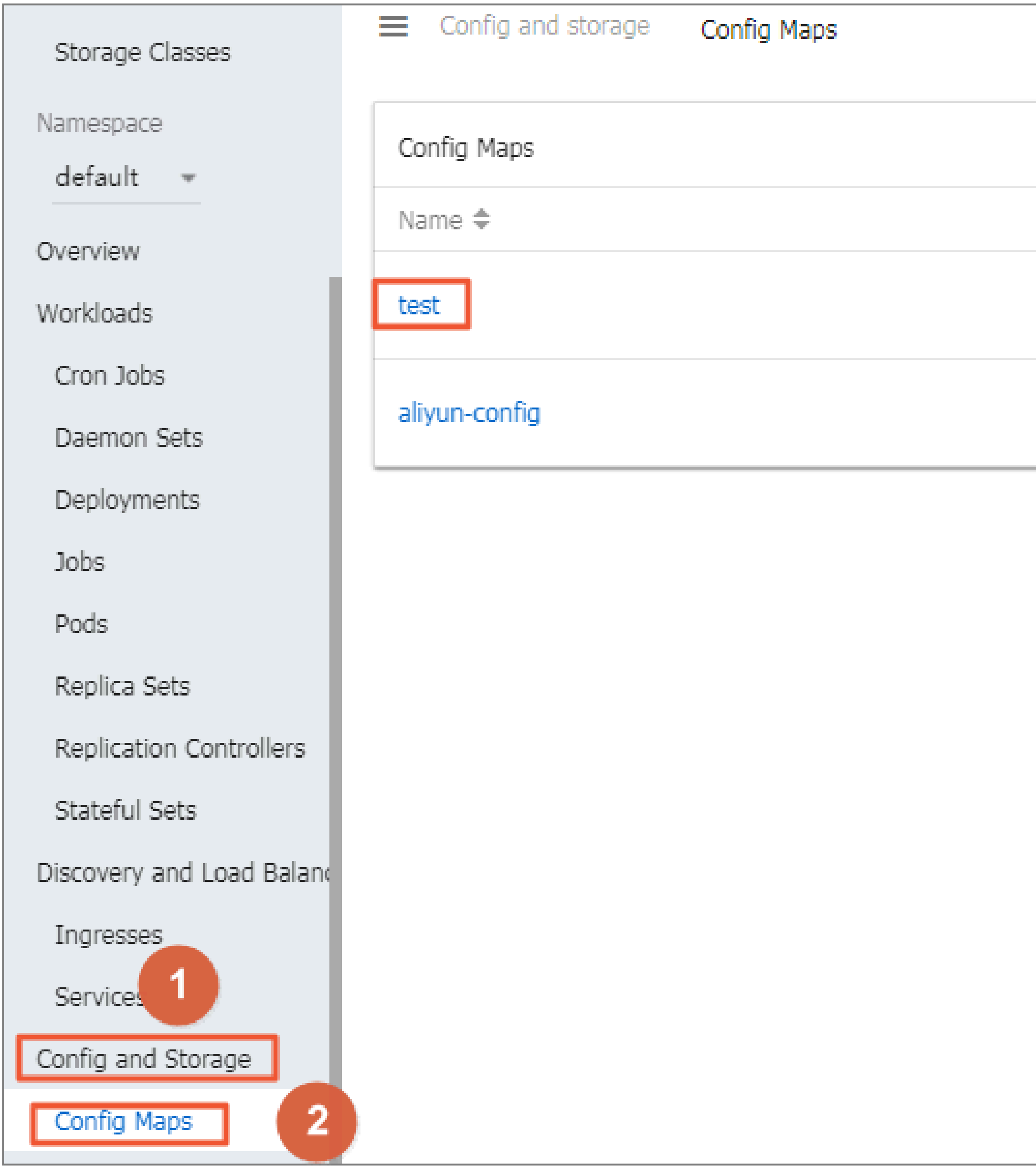
Modify a Config Map in the Kubernetes dashboard

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.
3. Find the target cluster. In the Action column, click Dashboard.



4. In the left-side navigation pane, select the target namespace, and then choose Config and Storage > Config Maps.

5. Find the target Config Map, and choose **Actions** > **View/edit YAML**.



6. In the displayed dialog box, modify the configurations, and then click UPDATE.

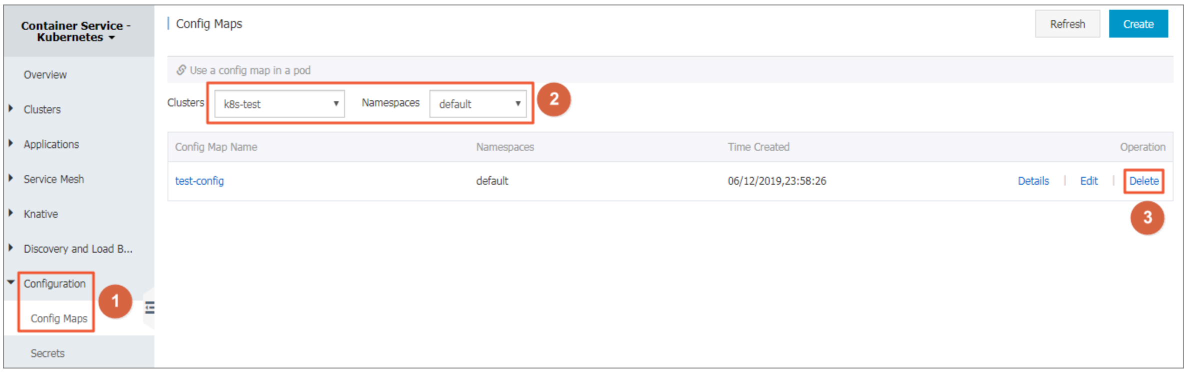
7.5 Delete a Config Map

This topic describes the two methods to delete a Config Map.

Delete a Config Map on the Config Maps page

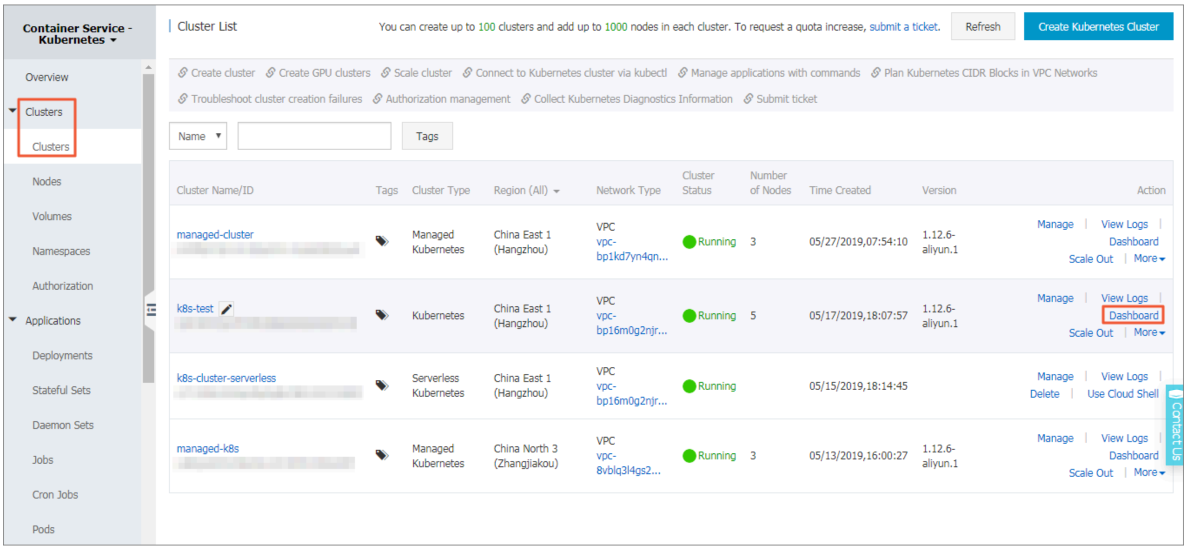
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Config Maps.

3. Select the target cluster and namespace, and find the target Config Map. Then, in the Operation column, click Delete.



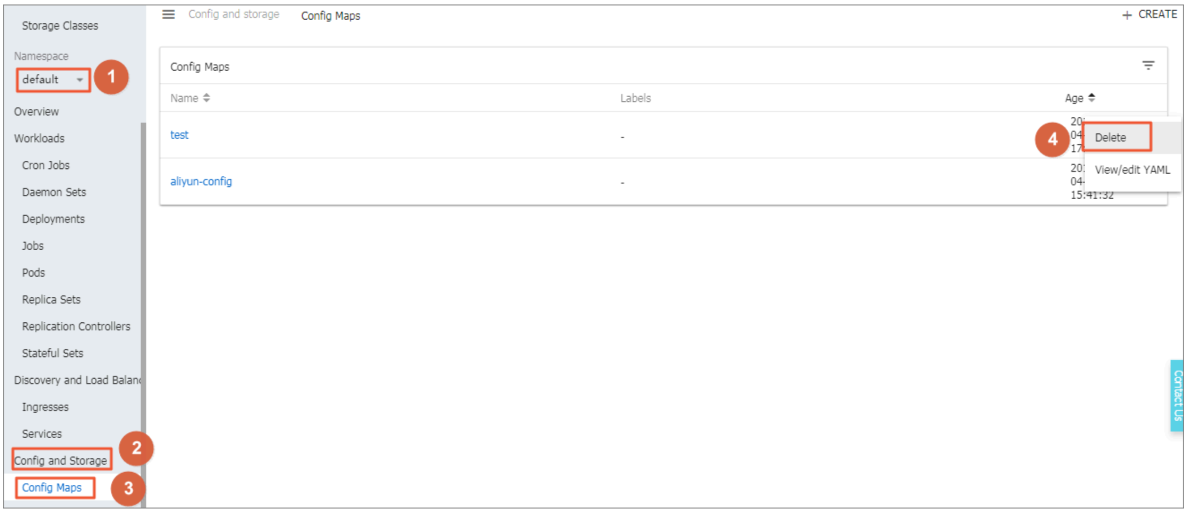
Delete a Config Map in the Kubernetes dashboard

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.
3. Find the target cluster. In the Action column, click Dashboard.



4. In the left-side navigation pane, select the target namespace, and then choose Config and Storage > Config Maps.

5. Find the target Config Map, and choose Actions > Delete.



6. In the displayed dialog box, click DELETE.

7.6 Create a Secret

This topic describes how to use the Container Service console to create a Secret.

Prerequisites

A Kubernetes cluster is created. For more information, see [Create a Kubernetes cluster](#).

Context

We recommend that you use Secrets for sensitive configurations in Kubernetes clusters, such as passwords and certificates.

Secrets have many types. For example:

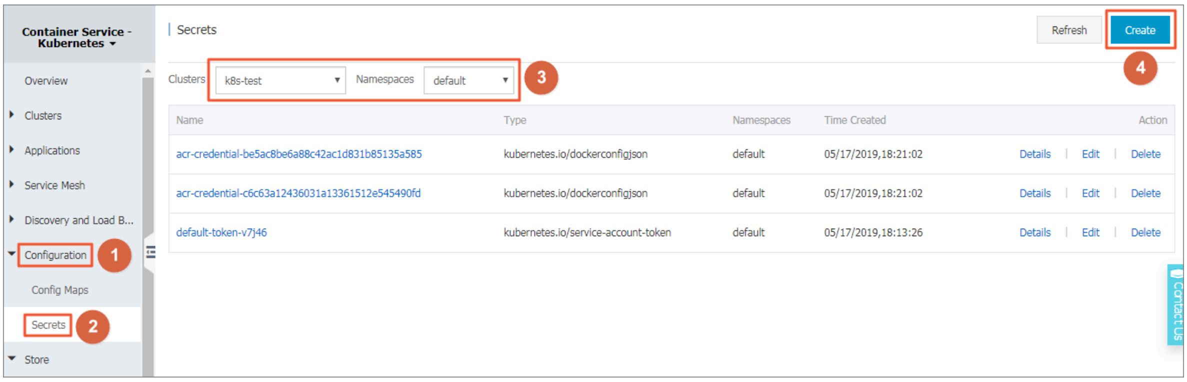
- Service Account: Automatically created by Kubernetes, which is used to access Kubernetes APIs and is automatically mounted to the pod directory `/run/secrets/kubernetes.io/serviceaccount`.
- Opaque: Secret in the base64 encoding format, which is used to store sensitive information such as passwords and certificates.

By default, you can only create secrets of the Opaque type in the Container Service console. Opaque data is of the map type, which requires the value to be in the base64 encoding format. Alibaba Cloud Container Service supports creating Secrets with one click and automatically encoding the clear data to base64 format.

You can also manually create Secrets by using command lines. For more information, see Kubernetes Secrets .

Procedure

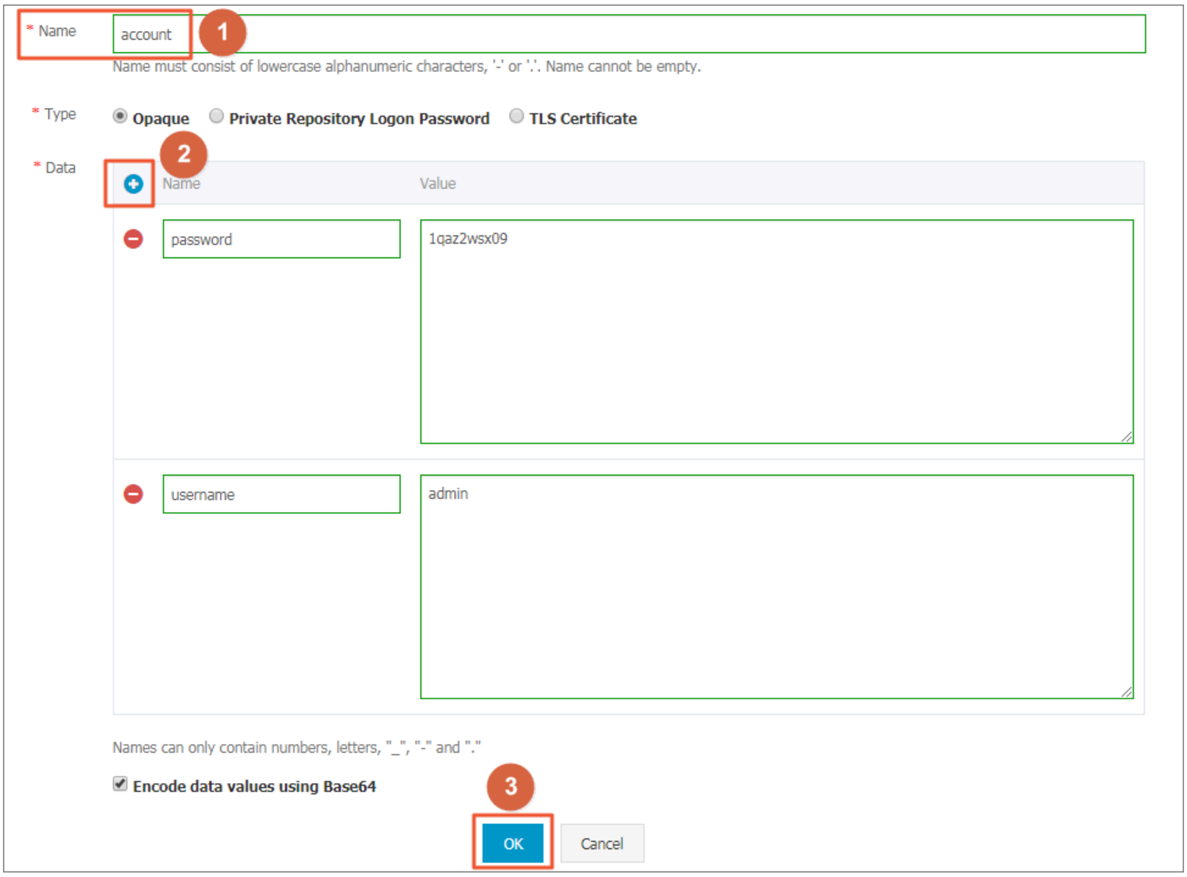
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Secrets.
3. Select the target cluster and namespace. Then, in the upper-right corner, click Create.



4. Complete the configurations to create a Secret.

Note:

To enter the clear data of the secret, select the Encode data values using Base64 check box.

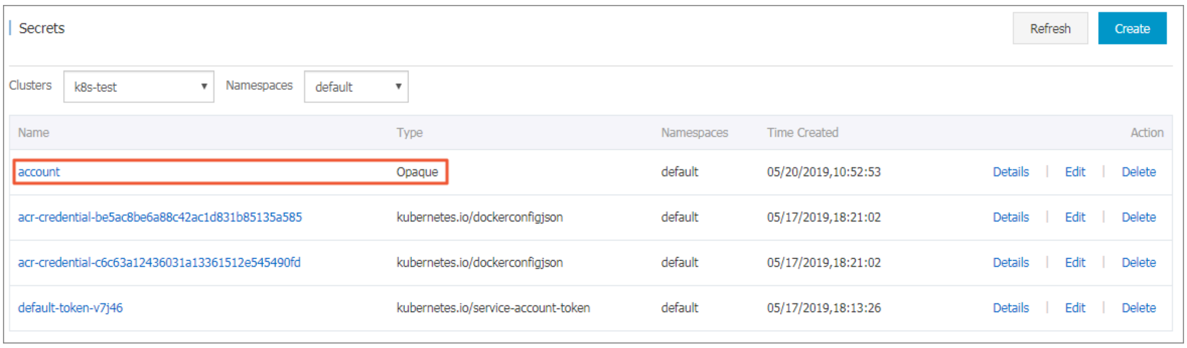


a. Name: Enter the Secret name, which must be 1–253 characters long, and can only contain lowercase letters, numbers, hyphens (-), and dots (.).

b. Configure the Secret data. Click the add icon next to Name and enter the name and value of the Secret, namely, the key-value pair. In this example, the Secret contains two values: `username : admin` and `passwrod : 1f2d1e2e67 df`.

c. Click OK.

5. The Secret page appears. You can view the created Secret in the Secret list.



7.7 Use a Secret in a pod

This topic describes how to use a Secret in a pod. You can use a Secret to configure volumes and environment variables in a pod.

Prerequisites

- A Secret named secret-test is created by using the following YAML template:

```
apiVersion :  v1
kind :  Secret
metadata :
  name :  secret - test
type :  Opaque
data :
  username :  admin
  password :  12345   # You   must   encode   your   password   by
  using   Base64 .
```

  For more information about how to create a secret, see Create a Secret.
- The Secret and pod share the same cluster and namespace.
- The Master node of your Kubernetes cluster is connected. For more information, see Connect to a Kubernetes cluster by using kubectl.

Use a Secret to configure pod volumes

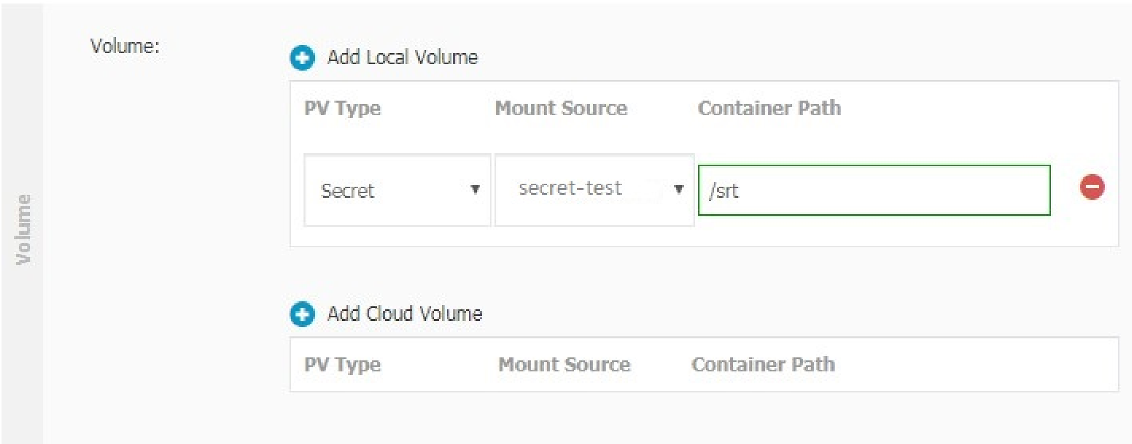
You can configure pod volumes by using either of the following methods:

- Method 1: Configure pod volumes by running the `kubectl  apply  - f  < example0 . yaml >` command.

  Note:
  You must replace the *example0 . yaml* file with the name of the target YAML file.

- Method 2: Configure pod volumes in the Container Service console.
  1. Log on to the Container Service console.
  2. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Deployment.
  3. In the upper-right corner, click Create from Image. For information about how to create a deployment application by using an image, see Create a deployment application by using an image.

     In the Volume section of the Container tab, click Add Local Volume, select `Secret` from the `PV Type` drop-down list, select `secret - test` from the `Mount Source` drop-down list, and set `Container Path` to the path where `secret - test` is saved.

A Secret can be used as a file in a pod. For example, the username and password of the Secret (secret-test) is saved to the `/ srt` directory as a file.

```
apiVersion :  v1
kind :  Pod
metadata :
  name :  pod0
spec : undefined   containers :
 -  name :  redis
    image :  redis
    volumeMoun  ts :
   -  name :  srt
      mountPath : "/ srt  "
      readOnly :  true
  volumes :
 -  name :  srt
    secret :
      secretName :  secret - test
```

Use a Secret to configure environment variables

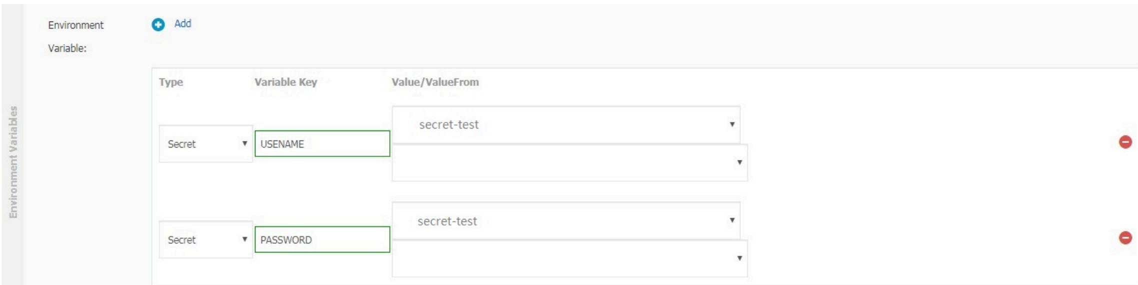
You can configure environment variables by using either of the following methods:

- Method 1: Configure environment variables by running the `kubectl apply -f < example1 . yaml >` command.

> **Note:**
> You must replace the *example1 . yaml* file with the name of the target YAML file.

- Method 2: Configure environment variables in the Container Service console.
  1. Log on to the Container Service console.
  2. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Deployment.
  3. In the upper-right corner, click Create from Image. For information about how to create a deployment application by using an image, see Create a deployment application by using an image.

     In the Environment Variable section of the Container tab, click Add, select `Secret` from the `Type` drop-down list, select `secret - test` from the `Value / ValueFrom` drop-down list, select the target keys, and enter a name for the variables.



For more information about Secrets, see Secrets.

7.8 View a Secret

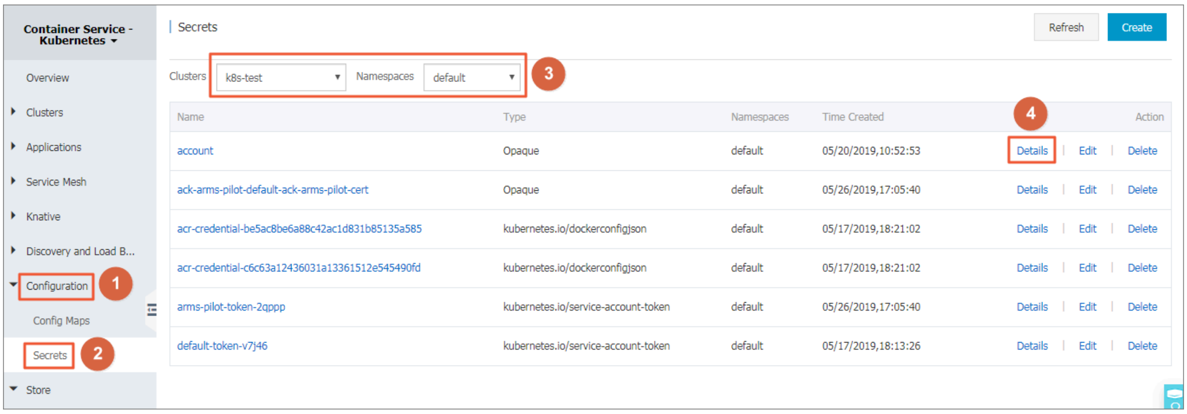
You can view the details of a created Secret in the Container Service console.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A Secret is created. For more information, see Create a Secret.

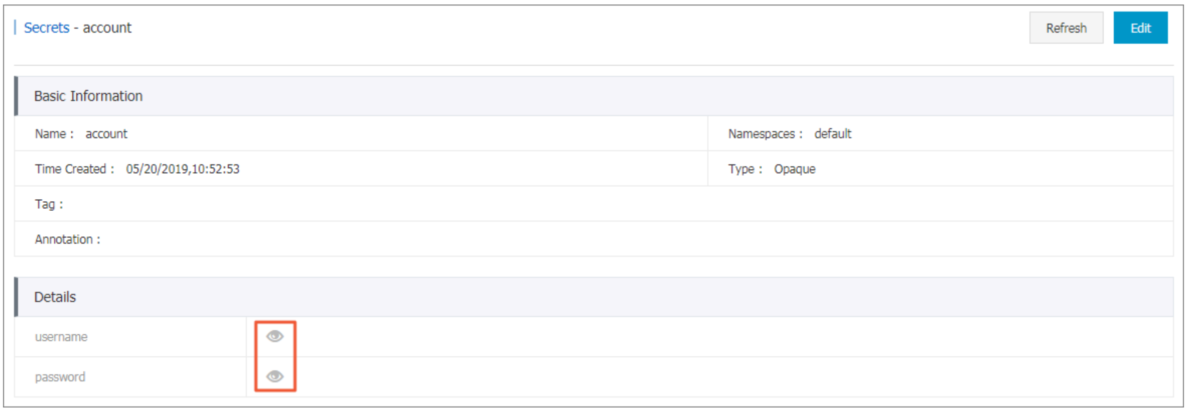
Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Secrets.
3. Select the target cluster and namespace, and find the target Secret. Then, in the Action column, click Details.



4. You can view the basic information of the secret, and the data that the secret contains.

   In the Details area, click the icon on the right of the data name to view the plain text.



7.9 Edit a Secret

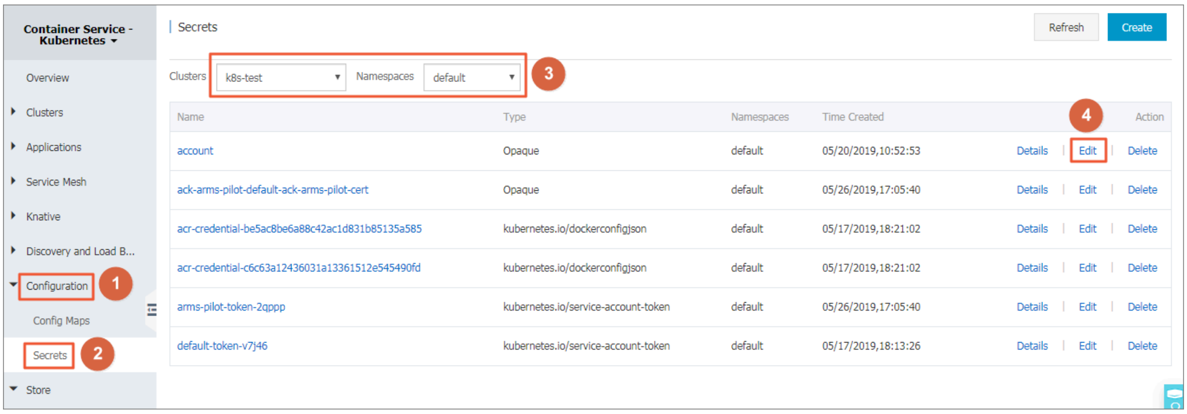
This topic describes how to edit a Secret in the Container Service console.

Prerequisites

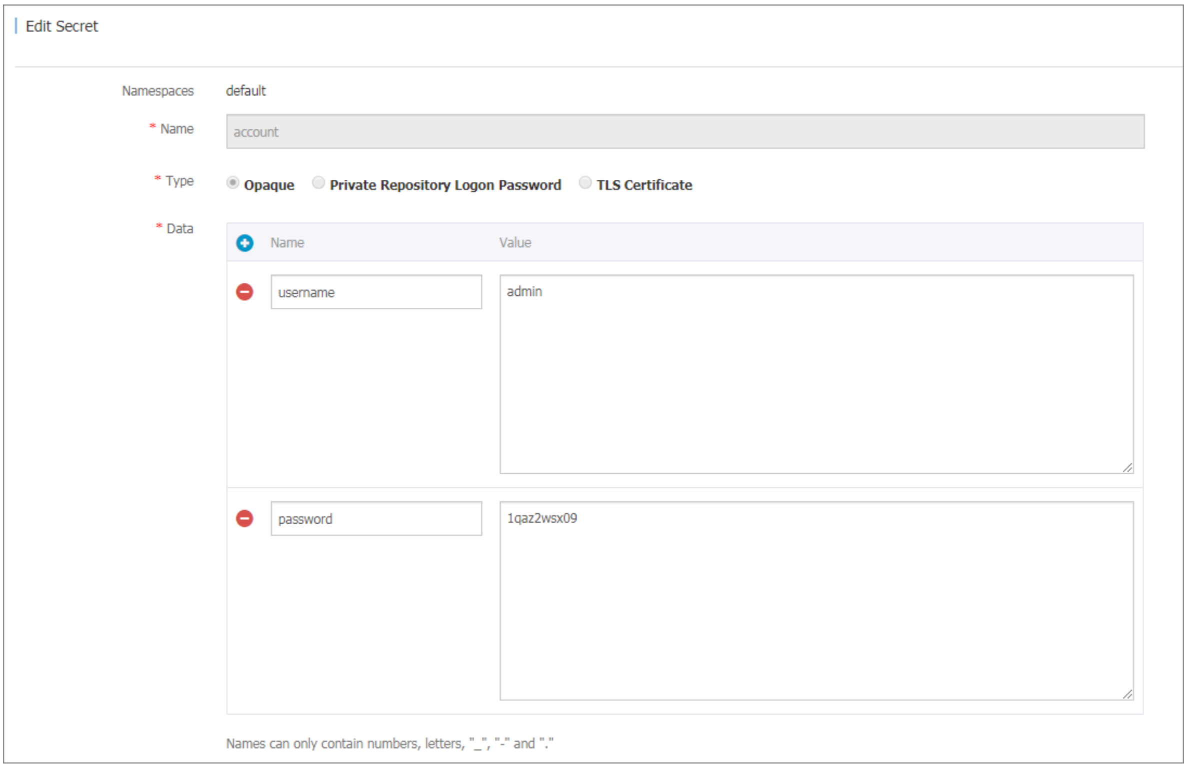
- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A Secret is created. For more information, see Create a Secret.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Secrets.
3. Select the target cluster and namespace, and find the target Secret. Then, in the Action column, click Edit.



4. Edit the Secret data on the Edit Secret page.



5. Click OK.

7.10 Delete a Secret

This topic describes how to delete a Secret in the Container Service console.

Prerequisites

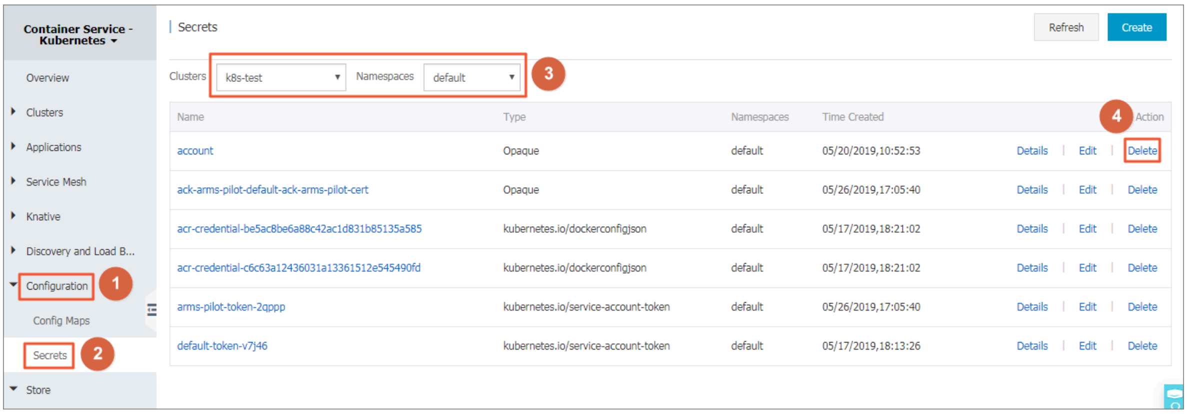
- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A Secret is created. For more information, see Create a Secret.

Context

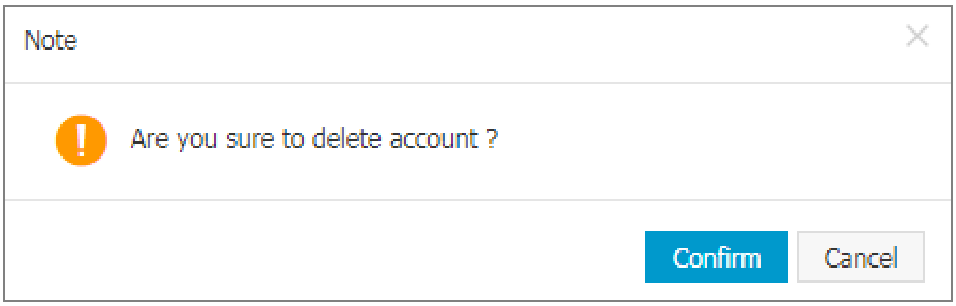
Do not delete the Secret generated when the cluster is created.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Configuration > Secrets.
3. Select the target cluster and namespace, and find the target Secret. Then, in the Action column, click Delete.



4. In the displayed dialog box, click Confirm.



8 Log management

8.1 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 Use Log Service to collect Kubernetes cluster logs, 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 Collect logs of a Kubernetes cluster by using Log-Pilot, Elasticsearch, and Kibana, you can easily build your own application log clusters.

8.2 View cluster logs

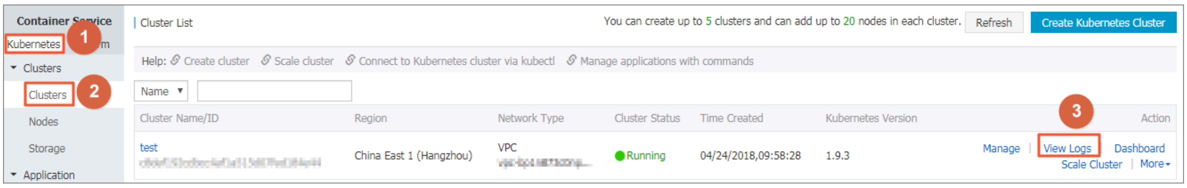
Context

You can view the cluster operation logs by using the simple log service of Container Service.

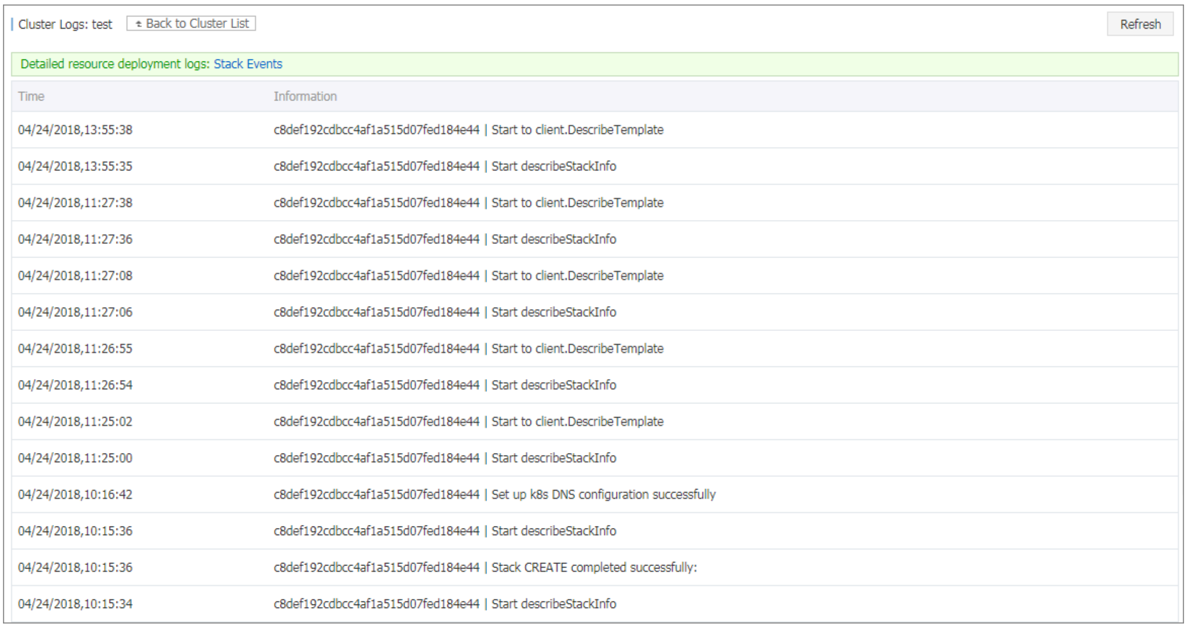
Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Clusters in the left-side navigation pane.

3. Click View Logs at the right of the cluster.



View the cluster operation information.



8.3 Use Log Service to collect Kubernetes cluster logs

Log Service is integrated with Kubernetes clusters of Alibaba Cloud Container Service. You can enable Log Service when creating a cluster to quickly collect container logs for the Kubernetes cluster, such as the standard output of the container and text files of the container.

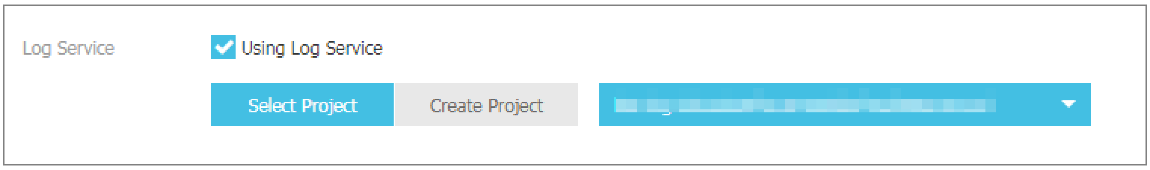
Enable Log Service when creating a Kubernetes cluster

If you have not created any Kubernetes clusters, follow steps in this section to enable Log Service:

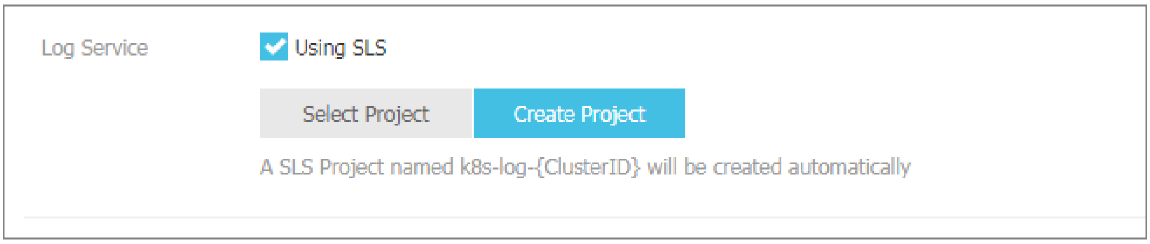
1. Log on to the Container Service console.
2. Click Clusters in the left-side navigation pane and click Create Kubernetes Cluster in the upper-right corner.
3. For how to configure a cluster on the creation page, see Create a Kubernetes cluster.

4. Drag to the bottom of the page and select the Using Log Service check box. The log plug-in will be installed in the newly created Kubernetes cluster.
5. When you select the Using Log Service check box, project options are displayed. A project is the unit in Log Service to manage logs. For more information about projects, see Project . Currently, two ways of using a project are available:

   - Select an existing project to manage collected logs.

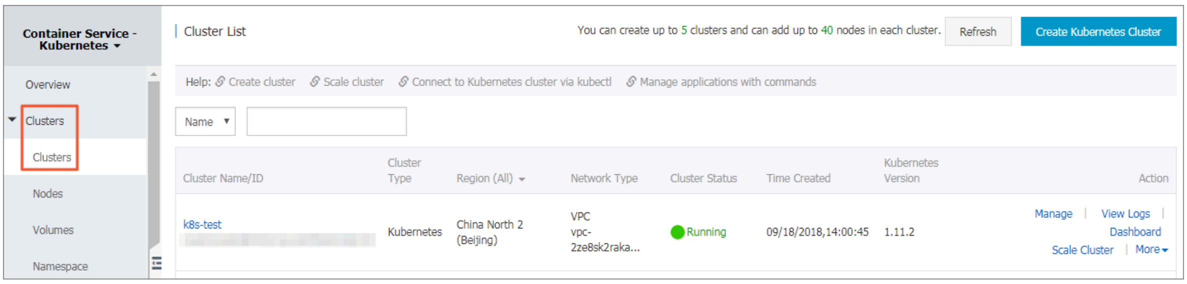
   

   - The system automatically creates a new project to manage collected logs. The project is automatically named `k8s - log -{ ClusterID }`, where ClusterID represents the unique identifier of the created Kubernetes cluster.

6. After you complete the configurations, click Create in the upper-right corner. In the displayed dialog box, click OK.

   After the cluster creation is completed, the newly created Kubernetes cluster is displayed on the cluster list page.

Manually install Log Service components in a created Kubernetes cluster

If you have created a Kubernetes cluster, following instructions in this section to use Log Service:

- Log Service components are not installed. Manually install the components.

- Log Service components are installed but in an earlier version. Upgrade the components. If you do not upgrade the components, you can only use the Log Service console or custom resource definition (CRD) to configure log collection.

Check the Log Service component version

1. Use Cloud Shell to connect to the target Kubernetes cluster.

   For more information, see Use kubectl commands in Cloud Shell to manage a Kubernetes cluster.
2. Run the following command to fast determine whether an upgrade or migration operation is required:

```
$ kubectl describe daemonsets - n kube - system logtail -
ds | grep ALICLOUD_L OG_DOCKER_ ENV_CONFIG
```

   - If `ALICLOUD_L OG_DOCKER_ ENV_CONFIG : true` is output, the components can be used directly without requiring upgrade or migration.
   - If other results are output, check the components further.
3. Run the following command to determine whether Helm is used to install the components.

```
$ helm get alibaba - log - controller | grep CHART
CHART : alibaba - cloud - log - 0 . 1 . 1
```

   - 0.1.1 in the output indicates the version of the Log Service components. Please use the version of 0.1.1 and later. If the version is too early, please see Upgrade Log Service components to upgrade the components. If you have used Helm to install the components of a valid version, you can skip next steps.
   - If no results are output, the components are installed by using Helm. But the DaemonSet installation method might be used. Follow the next step to check further.

4. DaemonSet can be an old one or a new one:

```
$ kubectl get daemonsets - n kube - system logtail
```

- If no result is output or `No resources found .` is output, the Log Service components are not installed. For information about the installation method, see Manually install Log Service components.
- If the correct result is output, an old DaemonSet is used to install the components which require upgrade. For information about upgrading the components, see Upgrade Log Service components.

Manually install Log Service components

1. Use Cloud Shell to connect to the target Kubernetes cluster.

   For more information, see Use kubectl commands in Cloud Shell to manage a Kubernetes cluster.
2. Run the `echo $ ALIBABA_CL OUD_ACCOUN T_ID` command in Cloud Shell to get the ID of your account in Alibaba Cloud.
3. Run the following command:

   Note:
   For this command, you need to specify the following parameters as required: `${ your_k8s_c luster_id }`, `{ your_ali_u id }`, and `{ your_k8s_c luster_reg ion_id }`.

```
wget https :// acs - logging . oss - cn - hangzhou . aliyuncs
. com / alicloud - k8s - log - installer . sh - O alicloud -
k8s - log - installer . sh ; chmod 744 ./ alicloud - k8s -
log - installer . sh ; ./ alicloud - k8s - log - installer . sh
 -- cluster - id ${ your_k8s_c luster_id } -- ali - uid ${
your_ali_u id } -- region - id ${ your_k8s_c luster_reg ion_id
}
```

   Parameter description

   - your_k8s_cluster_id: indicates you Kubernetes cluster ID.
   - your_ali_uid: indicates the ID of your account in Alibaba Cloud. It can be obtained in step 2.
   - your_k8s_cluster_region_id: indicates the region in which you Kubernetes cluster resides, which can be found in Regions and zones. For example, if the cluster resides in Hangzhou, the value of this parameter cn-hangzhou.

Installation example

```
[ root @ iZbp ****** biaZ  ~]#  wget   https :// acs - logging . oss
 - cn - hangzhou . aliyuncs . com / alicloud - k8s - log - installer
 . sh  - O   alicloud - k8s - log - installer . sh ;  chmod   744  ./
 alicloud - k8s - log - installer . sh ; ./ alicloud - k8s - log -
 installer . sh  -- cluster - id   c77a **************** 0106  -- ali
 - uid   19 ********* 19  -- region - id   cn - hangzhou
-- 2018 - 09 - 28   15 : 25 : 33 --   https :// acs - logging . oss -
 cn - hangzhou . aliyuncs . com / alicloud - k8s - log - installer .
 sh
 Resolving   acs - logging . oss - cn - hangzhou . aliyuncs . com ...
 118 . 31 . 219 . 217 ,  118 . 31 . 219 . 206
 Connecting   to   acs - logging . oss - cn - hangzhou . aliyuncs .
 com | 118 . 31 . 219 . 217 |: 443 ...  connected .
 HTTP   request   sent ,  awaiting   response ...  200   OK
 Length :  2273  ( 2 . 2K ) [ text / x - sh ]
 Saving   to : ‘ alicloud - k8s - log - installer . sh ’

 alicloud - k8s - log - installer . sh                         100
 %[==========================================================================
   2 . 22K   --. - KB / s      in   0s

 2018 - 09 - 28   15 : 25 : 33  ( 13 . 5   MB / s ) - ‘ alicloud - k8s
 - log - installer . sh ’  saved  [ 2273 / 2273 ]

-- 2018 - 09 - 28   15 : 25 : 33 --   http :// logtail - release -
 cn - hangzhou . oss - cn - hangzhou . aliyuncs . com / kubernetes /
 alibaba - cloud - log . tgz
 Resolving   logtail - release - cn - hangzhou . oss - cn - hangzhou .
 aliyuncs . com ...  118 . 31 . 219 . 49
 Connecting   to   logtail - release - cn - hangzhou . oss - cn -
 hangzhou . aliyuncs . com | 118 . 31 . 219 . 49 |: 80 ...  connected
 .
 HTTP   request   sent ,  awaiting   response ...  200   OK
 Length :  2754  ( 2 . 7K ) [ applicatio  n / x - gzip ]
 Saving   to : ‘ alibaba - cloud - log . tgz ’

 alibaba - cloud - log . tgz                                 100
 %[==========================================================================
   2 . 69K   --. - KB / s      in   0s

 2018 - 09 - 28   15 : 25 : 34  ( 79 . 6   MB / s ) - ‘ alibaba -
 cloud - log . tgz ’  saved  [ 2754 / 2754 ]

[ INFO ]  your   k8s   is   using   project  :  k8s - log -
 c77a92ec5a  3ce4e64a1b  f13bde1820  106
 NAME :    alibaba - log - controller
 LAST   DEPLOYED :  Fri   Sep   28   15 : 25 : 34   2018
 NAMESPACE :  default
 STATUS :  DEPLOYED

 RESOURCES :
==>  v1beta1 / CustomReso  urceDefini  tion
 NAME                                     AGE
 aliyunlogc  onfigs . log . alibabaclo  ud . com    0s

==>  v1beta1 / ClusterRol  e
 alibaba - log - controller    0s

==>  v1beta1 / ClusterRol  eBinding
 NAME                         AGE
 alibaba - log - controller    0s
```

```
==>  v1beta1 / DaemonSet
 NAME           DESIRED    CURRENT    READY    UP - TO - DATE
 AVAILABLE    NODE    SELECTOR    AGE
 logtail - ds    2          2          0        2             0
    < none >          0s

==>  v1beta1 / Deployment
 NAME                      DESIRED    CURRENT    UP - TO - DATE
 AVAILABLE    AGE
 alibaba - log - controller    1          1          1             0
          0s

==>  v1 / Pod ( related )
 NAME                                      READY    STATUS
    RESTARTS    AGE
 logtail - ds - 6v979                          0 / 1      ContainerC
 reating    0           0s
 logtail - ds - 7ccqv                          0 / 1      ContainerC
 reating    0           0s
 alibaba - log - controller - 84d8b6b8cf - nkrkx    0 / 1
 ContainerC  reating    0           0s

==>  v1 / ServiceAcc  ount
 NAME                      SECRETS    AGE
 alibaba - log - controller    1          0s


[ SUCCESS ]  install   helm   package  :  alibaba - log - controller
 success .
```

Upgrade Log Service components

If you have installed Log Service components of an early version through Helm or DaemonSet, upgrade or migrate the components as follows.

Note:

You need to use Cloud Shell to connect to the target Kubernetes cluster. For more information, see Use kubectl commands in Cloud Shell to manage a Kubernetes cluster.

Use Helm to upgrade Log Service components (recommended)

1. Run the following command to download the latest Helm package of Log Service components:

```
wget   http :// logtail - release - cn - hangzhou . oss - cn -
hangzhou . aliyuncs . com / kubernetes / alibaba - cloud - log .
tgz  - O   alibaba - cloud - log . tgz
```

2. Upgrade the components by using helm upgrade. The command is as follows:

```
helm  get  values  alibaba - log - controller  -- all  >  values
. yaml  &&  helm  upgrade  alibaba - log - controller  alibaba -
cloud - log . tgz  -- recreate - pods  - f  values . yaml
```

Use DaemonSet to upgrade Log Service components

You can upgrade Log Service components by modifying the DaemonSet template. If your image account is acs, upgrade the image tag to the latest version that can be viewed in Container Registry.If your image account is acs, upgrade the image tag to the latest version that can be viewed in Container Registry.

Note:

- If upgrading the tag has not enabled a rolling update of Logtail, you must manually remove the Logtail pod to trigger a Logtail update.
- You need to check whether Logtail runs on all nodes, including Master nodes. If Logtail does not run on all nodes, you must set tolerations for Logtail.

```
 toleration  s :
-  operator : " Exists "
```

For more information, see Latest Helm package configurations.

DaemonSet migrate

This upgrade method is applicable to the situation that you find the components are installed through the old DaemonSet when you check the Log Service component version. This method does not support configuring Log Service in Container Service. You can upgrade the components as follows:

1. At the end of the installation command, add a parameter which is the name of the project of Log Service used by your Kubernetes cluster.

   For example, if the project name is k8s-log-demo and the cluster ID is c12ba2028cxxxxxxxxxx6939f0b, then the installation command is:

```
wget  https :// acs - logging . oss - cn - hangzhou . aliyuncs
. com / alicloud - k8s - log - installer . sh  - O  alicloud -
k8s - log - installer . sh ;  chmod  744  ./ alicloud - k8s - log
- installer . sh ; ./ alicloud - k8s - log - installer . sh  --
cluster - id  c12ba2028c  xxxxxxxxxx  6939f0b  -- ali - uid  19
********* 19  -- region - id  cn - hangzhou  -- log - project
k8s - log - demo
```

2. After you complete the installation, log on the Log Service console.
3. After you complete the installation, log on the Log Service console.
4. In the Log service console, apply the history collection configuration of the project and Logstore to the new machine group `k8s - group -${ your_k8s_cluster_id }`.
5. After one minute, the history collection configuration is unbound from the history machine group.
6. When log collection is normal, you can delete the previously installed Logtail DaemonSet.

> **Note:**
> During the upgrade, some logs are duplicated. The CRD configuration management takes effect only for the configuration created by using CRD. The history configuration does not support the CRD management because the history configuration is created by using the non-CRD mode.

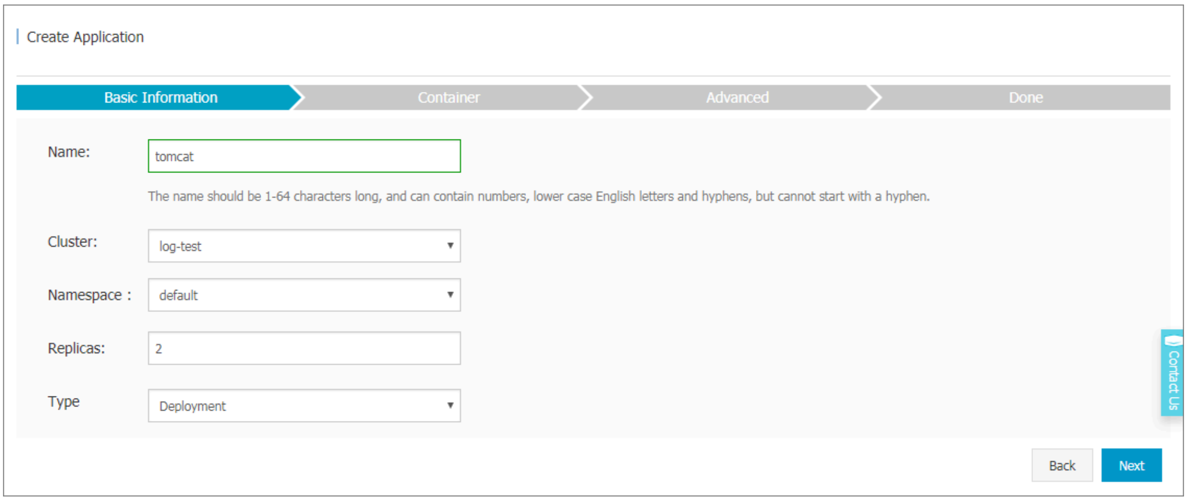
Configure Log Service when creating an application

Container Service allows you to configure Log Service to collect container logs when creating an application. Currently, you can use the console or a YAML template to create an application.

Create an application by using the console

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments. Then, click Create by Image in the upper-right corner.

3. Configure Name, Cluster, Namespace, Replicas, and Type, and then click Next.



4. On the Container page, select the Tomcat image and configure container log collection.

   The following describes only configurations related to Log Service. For information about other application configurations, see Create a deployment application by using an image.

5. Configure Log Service. Click the + sign to create a configuration which consists of a Logstore name and a log path.

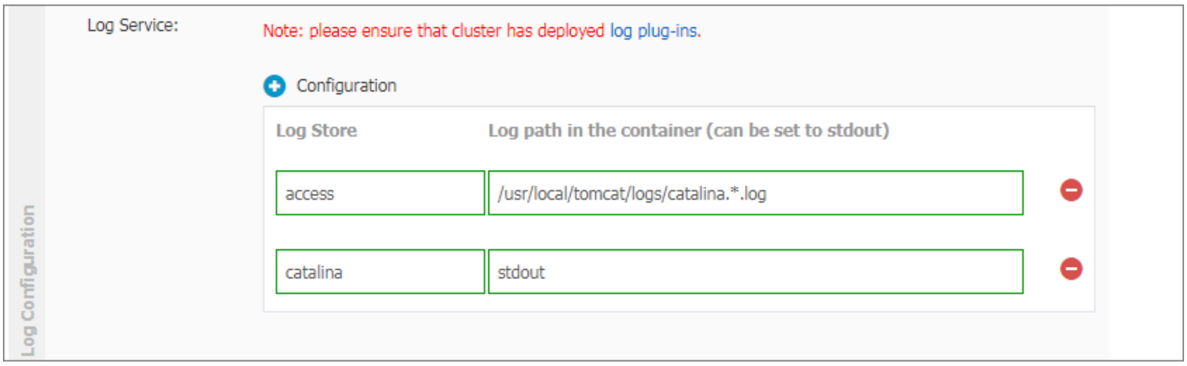
   - Logstore name: Specify a Logstore in which collected logs are stored. If your specified Logstore does not exist, the system automatically creates the Logstore in the project of Log Service with which the cluster is associated .

     Note:
     A Logstore name cannot contain underscores (_). You can use hyphens (-) instead.

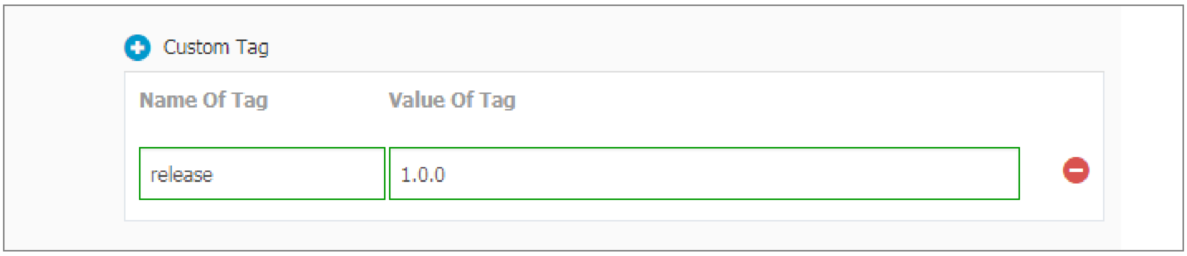
   - Log path: Specify the path where logs to be collected reside. For example, use `/usr/local/tomcat/logs/catalina.*.log` to collect text logs of tomcat.

     Note:
     If you specify the log path as stdout, the container standard output and standard error output will be collected.

   Each configuration is automatically created as a configuration for the corresponding Logstore. By default, the simple mode (by row) is used to collect logs. To use more collection modes, log on go to the Log Service console, and enter the corresponding project (prefixed with k8s-log by default) and Logstore to modify the configuration.



6. Custom tag. Click the + sign to create a new custom tag. Each custom tag is a key-value pair which will be added to collected logs. You can use a custom tag to mark container logs. For example, you can create a custom tag as a version number.



7. When you complete all the configurations of the container, click Next in the upper-right corner to perform further configurations. For more information, see Create a deployment application by using an image.

Create an application by using a YAML template

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments. Then, click Create by Template in the upper-right corner.
3. The syntax of the YAML template is the same as the Kubernetes syntax. To specify the collection configuration for the container, you need to use env to add collection configuration and custom tag for the container, and create corresponding volumeMounts and volumns. The following is a simple pod example:

```
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   environmen  t   variables   ###########
   -  name :  aliyun_log  s_log - stdout
      value :  stdout
   -  name :  aliyun_log  s_log - varlog
      value : / var / log /*. log
   -  name :  aliyun_log  s_mytag1_t  ags
      value :  tag1 = v1
   ###############################
   #########  Configure   vulume   mount  ###########
    volumeMoun  ts :
   -  name :  volumn - sls - mydemo
      mountPath : / var / log
  volumes :
 -  name :  volumn - sls - mydemo
    emptyDir : {}
 ###############################
```

- Configure three parts in order based on your needs.
- In the first part, use environment variables to create your collection configuration and custom tag. All environment variables related to configuration are prefixed with `aliyun_log  s_`.
- Rules for creating the collection configuration are as follows:

```
-  name :  aliyun_log  s_ { Logstore   name }
   value : { log   path }
```

  In the example, create two collection configurations. The `aliyun_log  s_log - stdout` env creates a configuration that contains a Logstore named log-stdout and the log path of stdout. The standard output of the container is collected and stored to the Logstore named log-stdout.

  Note:
  A Logstore name cannot contain underscores (_). You can use hyphens (-) instead.

- Rules for creating a custom tag are as follows:

```
-  name :  aliyun_log  s_ { a   name   without  ' _ '} _tags
```

```
value : { Tag   name }={ Tag   value }
```

After a tag is configured, when logs of the container are collected, fields corresponding to the tag are automatically attached to Log Service.

- If you specify a non-stdout log path in your collection configuration, create corresponding volumnMounts in this part.

  In the example, the `/ var / log /*. log` log path is added to the collection configuration, therefore, the `/ var / log` volumeMounts is added.

4. When you complete a YAML template, click DEPLOY to deliver the configurations in the template to the Kubernetes cluster to execute.

Set environment variables to control log collection

The environment variables of containers support multiple advanced parameters, which that can be used to control the collection of logs. The following table details these parameters.

Parameter	Description	Example	Remarks

`aliyun_logs_{key}`	· Required. This parameter is used to specify the name of a log collection configuration. The value of `{key}` must start with a letter, and be followed by letters, numbers, and hyphens (-). It cannot contain underlines (_).
	· If you do not specify a Logstore by using the optional parameter `aliyun_logs_{key}_logstore`, logs are automatically collected to the Logstore named `{key}`.
	· To collect the standard output of containers, set this parameter as stdout. To collect logs of a specific path, set this parameter as the path.
		· `- name : aliyun_logs_catalina`
		`stdout`
		· `- name : aliyun_logs_access-log`
		`/var/log/nginx/access.log`
			· By default, the simple collection mode is used to collect logs. If you want to parse collected logs, we recommend that you use the Log Service console to follow the procedure described in Container text logs, Container stdout, or Configure Kubernetes log collection on CRD.
			· The value of `{key}` must be unique within the target Kubernetes cluster.

`aliyun_logs_{key}_tags`	Optional. This parameter is used	`- name : aliyun_log`	-

Set the `aliyun_log s_ { key } _logstore` parameter to implement this task. For example, to collect the stdout logs of two applications to a Logstore named `stdout - logstore`, use this parameter to set the environment variables of two applications.

Set the environment variables of Application 1 as follows:

```
######### set environmen t variables ###########
    - name : aliyun_log s_app1 - stdout
      value : stdout
    - name : aliyun_log s_app1 - stdout_log store
      value : stdout - logstore
```

Set the environment variables of Application 2 as follows:

```
######### set environmen t variables ###########
    - name : aliyun_log s_app2 - stdout
      value : stdout
    - name : aliyun_log s_app2 - stdout_log store
      value : stdout - logstore
```

Scenario 2: Collect the log data of each application to an exclusive project

Note:
This means that a project stores only the log data of one specific application.

To implement this task, follow these steps to configure each application:

1. In each project, create a machine group and customize an identifier for the machine group.

   Note:
   The identifier name is in the format of k8s-group-{cluster-id}.
2. For the environment variables of each application, set the following parameter: `project`, `logstore`, and `machinegro up`.

   Note:
   The value of the `machinegro up` parameter is the name of the machine group that you create in step 1.

   The following shows the example settings for environment variables of an application:

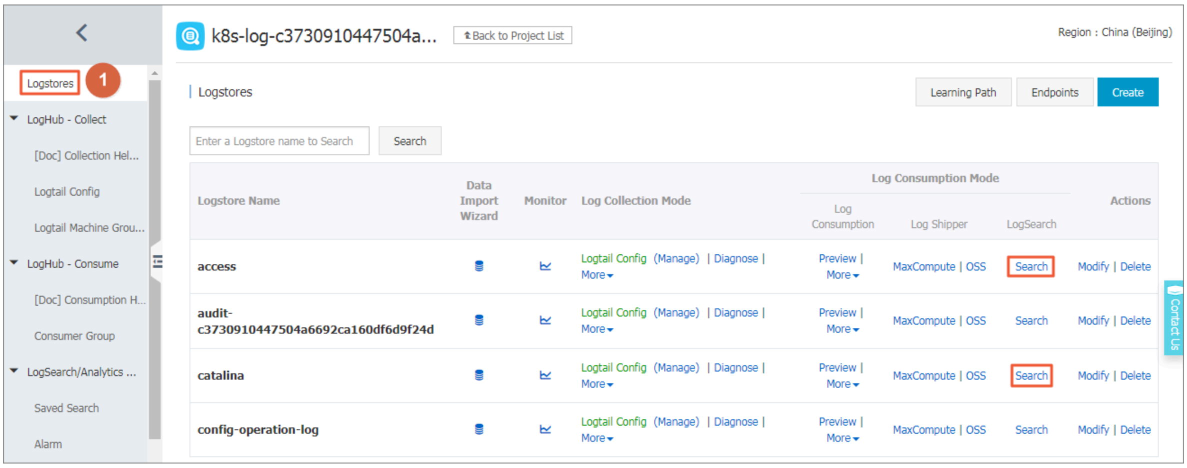
```
######### set environmen t variables ###########
    - name : aliyun_log s_app1 - stdout
      value : stdout
```

```
    - name : aliyun_log s_app1 - stdout_pro ject
      value : app1 - project
    - name : aliyun_log s_app1 - stdout_log store
      value : app1 - logstore
    - name : aliyun_log s_app1 - stdout_mac hinegroup
      value : app1 - machine - group
```

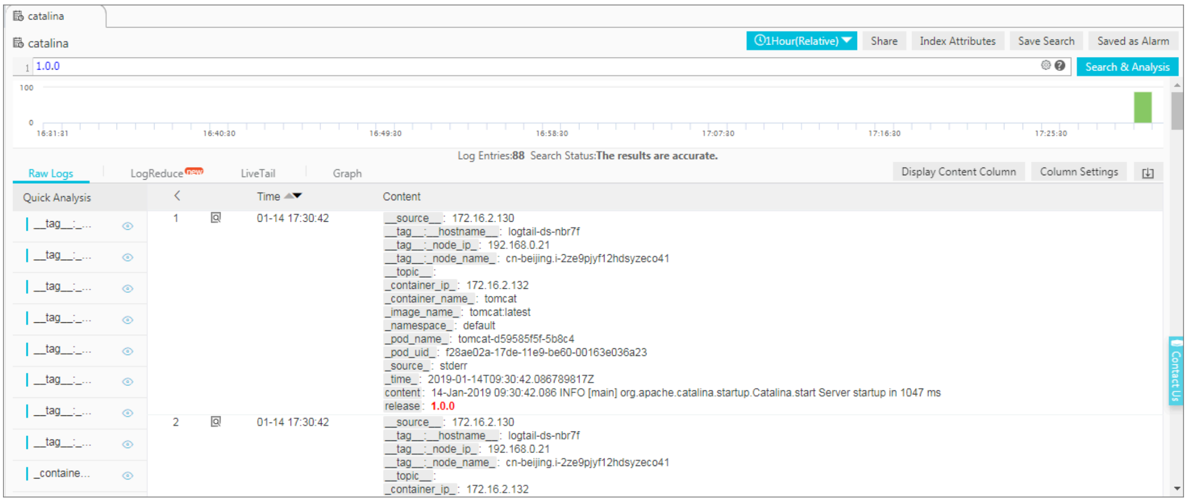
View logs

In this example, view logs of the tomcat application created in the console. After you complete the application configuration, logs of the tomcat application are collected and stored to Log Service. You can view your logs as follows:

1. Log on to the Log Service console.
2. Log on to the Log Service console.
3. In the console, select the project (k8s-log-{Kubernetes cluster ID} by default) corresponding to the Kubernetes cluster.
4. In the Logstore list, locate the Logstore specified in your configuration and click Search.



5. In this example, on the log search page, you can view the standard output logs of the tomcat application and text logs in the container, and you can find your custom tag is attached to log fields.



More information

1. By default, the system use the simple mode to collect your data, that is, to collect data by row without parsing. To perform more complex configurations, see the following Log Service documents and log on to the Log Service console to modify configurations.
   - Container text logs
   - Container stdout
2. In addition to configuring log collection through the console, you can also directly collect logs of the Kubernetes cluster through the CRD configuration. For more information, see Configure Kubernetes log collection on CRD.
3. For troubleshooting exceptions, see Troubleshoot collection errors.

8.4 Collect logs of a Kubernetes cluster by using Log-Pilot, Elasticsearch, and Kibana

This topic describes how to collect logs of a Kubernetes cluster by using Log-Pilot, Elasticsearch, and Kibana.

Requirements for logs of distributed Kubernetes clusters always bother developers . This is mainly because of the characteristics of containers and the defects of log collection tools.

- Characteristics of containers:
  - Many collection targets: The characteristics of containers cause the number of collection targets is large, which requires to collect the container logs and container stdout. Currently, no good tool can collect file logs from containers dynamically. Different data sources have different collection softwares. However, no one-stop collection tool exists.
  - Difficulty caused by auto scaling: Kubernetes clusters are in the distributed mode. The auto scaling of services and the environment brings great difficulty to log collection. You cannot configure the log collection path in advance, the same as what you do in the traditional virtual machine (VM) environment. The dynamic collection and data integrity are great challenges.
- Defects of current log collection tools:
  - Lack the capability to dynamically configure log collection: The current log collection tools require you to manually configure the log collection method and path in advance. These tools cannot dynamically configure the log collection because they cannot automatically detect the lifecycle changes or dynamic migration of containers.
  - Log collection problems such as logs are duplicate or lost: Some of the current log collection tools collect logs by using the tail method. Logs may be lost in this way. For example, the application is writing logs when the log collection tool is being restarted. Logs written during this period may be lost. Generally, the conservative solution is to collect logs of 1 MB or 2 MB previous to the current log by default. However, this may cause the duplicate log collection.
  - Log sources without clear marks: An application may have multiple containers that output the same application logs. After all the application logs are collected to a unified log storage backend, you cannot know a log is generated on which application container of which node when querying logs.

This document introduces Log-Pilot, a tool to collect Docker logs, and uses the tool together with Elasticsearch and Kibana to provide a one-stop solution to log collection problems in the Kubernetes environment.

Introduction on Log-Pilot

Log-Pilot is an intelligent tool used to collect container logs, which not only collects container logs and outputs these logs to multiple types of log storage backends efficiently and conveniently, but also dynamically discovers and collects log files from containers.

Log-Pilot uses declarative configuration to manage container events strongly and obtain the stdout and file logs of containers, which solves the problem of auto scaling. Besides, Log-Pilot has the functions of automatic discovery, maintenance of checkpoint and handle, and automatic tagging for log data, which effectively deals with the problems such as dynamic configuration, duplicate logs, lost logs, and log source marking.

Log-Pilot is an open-source project in GitHub.

Declarative configuration for container logs

Log-Pilot supports managing container events, can dynamically listen to the event changes of containers, parse the changes according to the container labels, generate the configuration file of log collection, and then provide the file to collection plug-in to collect logs.

For Kubernetes clusters, Log-Pilot can dynamically generate the configuration file of log collection according to the environment variable `aliyun_log s_ $ name = $ path`. This environment variable contains the following two variables:

- One variable is $name, a custom string which indicates different meanings in different scenarios. In this scenario, $name indicates index when collecting logs to Elasticsearch.
- The other is $path which supports two input modes, stdout and paths of log files within containers, respectively corresponding to the standard output of logs and log files within containers.
  - Stdout indicates to collect standard output logs from containers. In this example, to collect Tomcat container logs, configure the label `aliyun . logs . catalina = stdout` to collect standard output logs of Tomcat.
  - The path of a log file within a container also supports wildcards. To collect logs within the Tomcat container, configure the environment variable `aliyun_log s_access =/ usr / local / tomcat / logs /*. log`. To not use the keyword aliyun, you can use the environment variable PILOT_LOG_PREFIX, which is also provided by Log-Pilot, to specify the prefix of your declarative log configuration. For example, `PILOT_LOG_ PREFIX : " aliyun , custom "`.

Besides, Log-Pilot supports multiple log parsing formats, including none, JSON, CSV, Nginx, apache2, and regexp. You can use the `aliyun_logs_$name_format=<format>` label to tell Log-Pilot to use what format to parse logs when collecting logs.

Log-Pilot also supports custom tags. If you configure `aliyun_logs_$name_tags="K1=V1,K2=V2"` in the environment variable, K1=V1 and K2=V2 are collected to log output of the container during the log collection. Custom tags help you tag the log generation environment for convenient statistics, routing, and filter of logs.

Log collection mode

In this document, deploy a Log-Pilot on each machine and collect all the Docker application logs from the machines.

Compared with deploying a logging container on each pod, the most obvious advantage of this solution is less occupied resources. The larger the cluster scale is, the more obvious the advantage is. This solution is also recommended in the community.

Prerequisites

- A Kubernetes cluster is created with ACK. For more information, see Create a Kubernetes cluster.
- An Elasticsearch instance is created. For more information, see Activate Alibaba Cloud Elasticsearch.
- The Elasticsearch cluster can be accessed through the intranet or Internet.

Procedure overview

- Step 1: Enable the Auto Indexing feature for the target Elasticsearch instance
- Step 2: Deploy the Log-Pilot components on your Kubernetes cluster
- Step 3: Collect logs of the target application

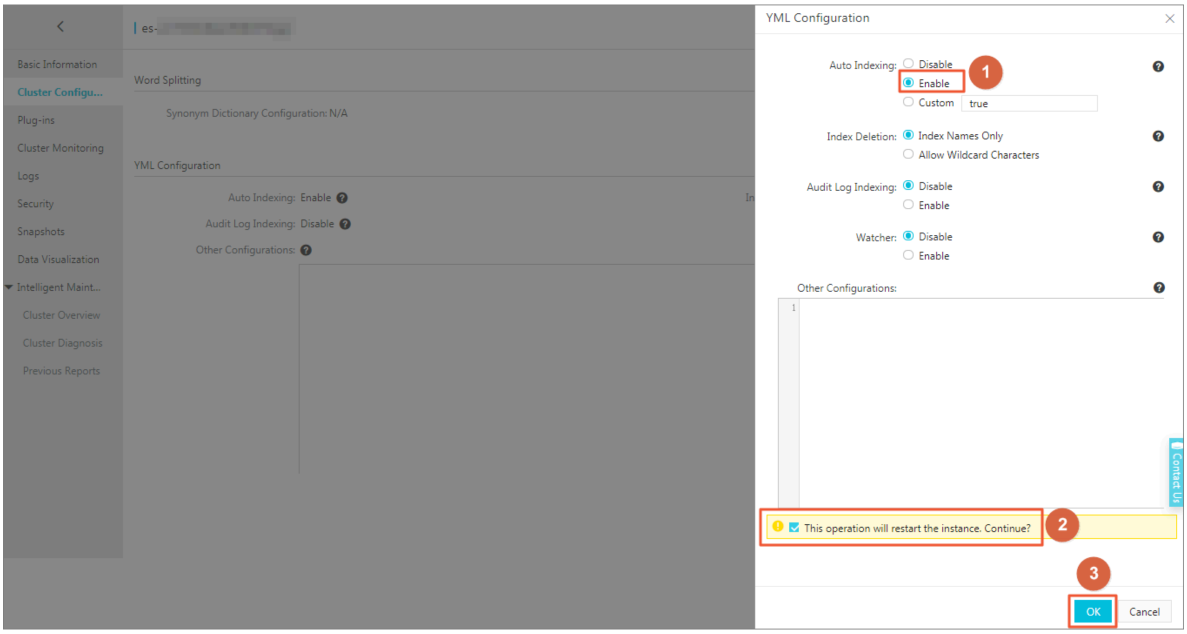
Procedure

Step 1: Enable the Auto Indexing feature for the target Elasticsearch instance

> **Note:**
> Alibaba Cloud Elasticsearch by default disables the Auto Indexing feature. However, to automatically collect container logs, Log-Pilot automatically creates indexes

according to the configurations of log collection. Therefore, you must enable the Auto Indexing feature.

1. Log on to the Elasticsearch console.
2. Find the target Elasticsearch instance, and then, in the Actions column, click Manage.
3. In the left-side navigation pane, click Cluster Configuration. Then, on the right side of the page, click Modify Configuration.
4. On the right of `Auto Indexing`, select the `Enable` radio button. Then, on the bottom of the page, select the check box This operation will restart the instance. Continue?, and click OK.

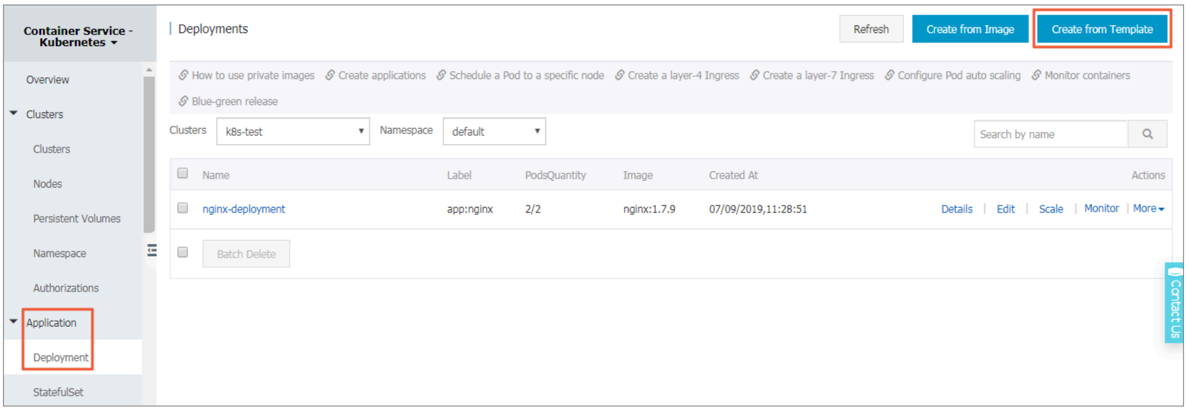


Step 2: Deploy the Log-Pilot components on your Kubernetes cluster

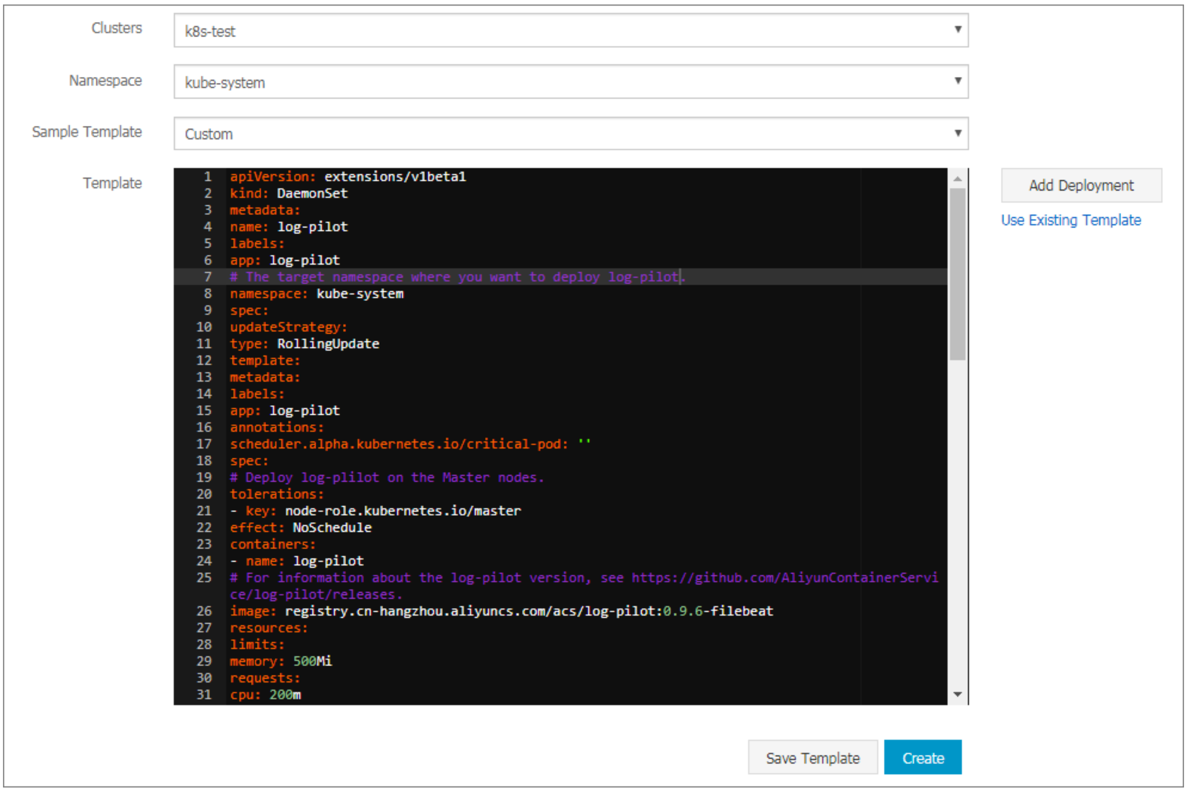
To save resources for cluster nodes, deploy the Log-Pilot components as a DaemonSet object on each node of the target Kubernetes cluster.

1. Log on to the Container Service console.

2. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Deployment.



3. In the upper-right corner, click Create from Template.
4. Select the target cluster where you want to deploy the Log-Pilot component, and select the kube-system namespace.
5. Select Custom from the Sample Template drop-down list, copy the follow code into the Template area, and click Create.



```
apiVersion :  extensions / v1beta1
kind :  DaemonSet
metadata :
name :  log - pilot
labels :
app :  log - pilot
#  The   target   namespace   where   you   want   to   deploy   log
 - pilot .
```

```
 namespace :  kube - system
 spec :
 updateStra  tegy :
 type :  RollingUpd  ate
 template :
 metadata :
 labels :
 app :  log - pilot
 annotation  s :
 scheduler . alpha . kubernetes . io / critical - pod : ''
 spec :
#  Deploy   log - plilot   on   the   Master   nodes .
 toleration  s :
-  key :  node - role . kubernetes . io / master
 effect :  NoSchedule
 containers :
-  name :  log - pilot
#  For   informatio  n   about   the   log - pilot   version ,  see
   https :// github . com / AliyunCont  ainerServi  ce / log - pilot
 / releases
 image :  registry . cn - hangzhou . aliyuncs . com / acs / log -
 pilot : 0 . 9 . 6 - filebeat
 resources :
 limits :
 memory :  500Mi
 requests :
 cpu :  200m
 memory :  200Mi
 env :
-  name : " NODE_NAME "
 valueFrom :
 fieldRef :
 fieldPath :  spec . nodeName
-  name : " LOGGING_OU  TPUT "
 value : " elasticsea  rch "
#  Make   sure   that   your   Kubernetes   cluster   can   access
 the   target   Elasticsea  rch   instance .
-  name : " ELASTICSEA  RCH_HOSTS "
 value : "{ es_endpoin  t }:{ es_port }"
#  Set   the   permission  s   to   access   the   target
 Elasticsea  rch   instance .
-  name : " ELASTICSEA  RCH_USER "
 value : "{ es_usernam  e }"
-  name : " ELASTICSEA  RCH_PASSWO  RD "
 value : "{ es_passwor  d }"
 volumeMoun  ts :
-  name :  sock
 mountPath : / var / run / docker . sock
-  name :  root
 mountPath : / host
 readOnly :  true
-  name :  varlib
 mountPath : / var / lib / filebeat
-  name :  varlog
 mountPath : / var / log / filebeat
-  name :  localtime
 mountPath : / etc / localtime
 readOnly :  true
 livenessPr  obe :
 failureThr  eshold :  3
 exec :
 command :
- / pilot / healthz
 initialDel  aySeconds :  10
```

```
periodSeco  nds :  10
successThr  eshold :  1
timeoutSec  onds :  2
securityCo  ntext :
capabiliti  es :
add :
-  SYS_ADMIN
terminatio  nGracePeri  odSeconds :  30
volumes :
-  name :  sock
hostPath :
path : / var / run / docker . sock
-  name :  root
hostPath :
path : /
-  name :  varlib
hostPath :
path : / var / lib / filebeat
type :  DirectoryO  rCreate
-  name :  varlog
hostPath :
path : / var / log / filebeat
type :  DirectoryO  rCreate
-  name :  localtime
hostPath :
path : / etc / localtime
```

- `{ es_endpoin t }`: indicates the address used to access the target Elasticsearch instance.

  > Note:
  > The address of the target Elasticsearch is shown on the Basic Information page of the Elasticsearch console.

  - If your Kubernetes cluster and the target Elasticsearch instance are deployed within the same VPC, set this parameter as the Internal Network Address of the Elasticsearch instance.
  - If your Kubernetes cluster and the target Elasticsearch instance are deployed in different VPCs, set this parameter as the Public Network Address of the Elasticsearch instance.
- `{ es_port }`: indicates the port used to access the target Elasticsearch instance. We recommend that you set it as 9200.
- `{ es_usernam e }`: indicates the user name used to access the target Elasticsearch instance. The default is elastic.
- `{ es_passwor d }`: indicates the password used to access the target Elasticsearch instance. It is the password that you set when you created the Elasticsearch instance.

6. Click Kubernetes Dashboard to view the progress.
7. In the left-side navigation pane, select kube-system from the Namespace drop-down list. Then, choose Workloads > Pods to verify that the Log-Pilot components are in the Running status.

> Note:
> You can also use kubectl to access your Kubernetes cluster, and then run the following commands to view the status of the Log-Pilot components:

```
~  kubectl   apply  - f   log - pilot . yml
 daemonset . extensions  " log - pilot "  created
~  kubectl  - n   kube - system   get   pod  |  grep   log - pilot
 log - pilot - 458nj   1 / 1   Running   0   23s
 log - pilot - 8ld4n   1 / 1   Running   0   23s
 log - pilot - b4kqv   1 / 1   Running   0   23s
 log - pilot - gd588   1 / 1   Running   0   23s
 log - pilot - k2ttk   1 / 1   Running   0   23s
```

Step 3: Collect logs of the target application

A Tomcat application is deployed in the Kubernetes cluster as the example to test whether logs of the application can be collected, indexed, and displayed.

> Note:
> In this example, the application type of the Tomcat application is set as Deployment. The configurations are also applicable to the application type of StatefulSet.

1. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Deployment.
2. In the upper-right corner, click Create from Template.
3. Select the same cluster as that you selected to deploy the Log-Pilot components, and select the target namespace.
4. Select Custom from the Sample Template drop-down list, copy the follow code into the Template area, and click Create.

```
 apiVersion :  v1
 kind :  Pod
 metadata :
 name :  tomcat
 spec :
 containers :
-  name :  tomcat
 image : " tomcat : 7 . 0 "
 env :
#  1 .  stdout   is   a   fixed   keyword   that   indicates   to
 collect   standard   output   logs .
```

```
# 2 . Collect the standard output logs to the
 catalina index of the target Elasticsea rch instance .
- name : aliyun_log s_catalina
 value : " stdout "
# 1 . Collect container logs what are in file format
 . Wildcard characters are supported .
# 2 . Collect the logs to the access index of the
   target Elasticsea rch instance .
- name : aliyun_log s_access
 value : "/ usr / local / tomcat / logs / catalina .*. log "
# To set the path of the container logs that are
   in the file format , set the emptyDir parameter .
 volumeMoun ts :
- name : tomcat - log
 mountPath : / usr / local / tomcat / logs
 volumes :
- name : tomcat - log
 emptyDir : {}
```

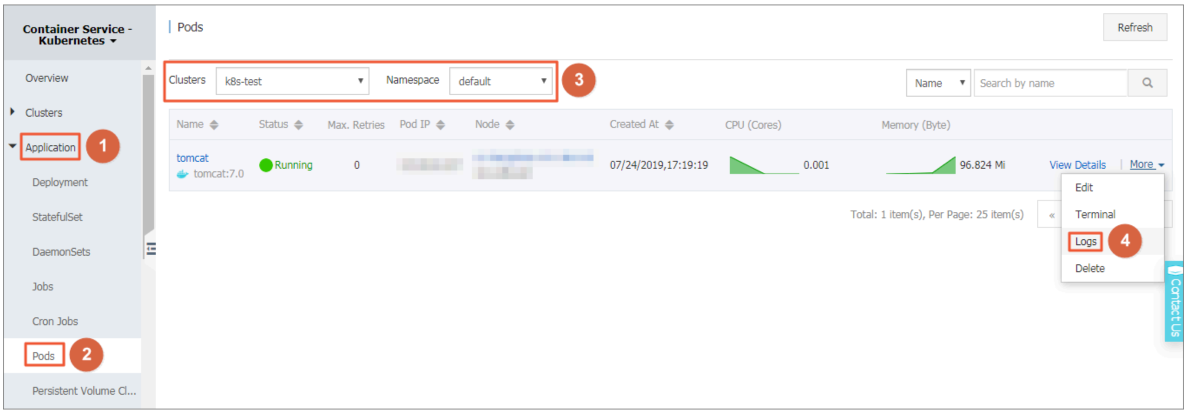
By customizing environment variables in a pod, the preceding orchestration file dynamically generates the configuration file to collect logs. The following settings of environment variables:

- `aliyun_log s_catalina = stdout` indicates to collect the stdout logs of the target containers.
- `aliyun_log s_access =/ usr / local / tomcat / logs / catalina .*. log` indicates to collect the logs (in the file format) that meet the following requirements:
  - The logs are generated in the directory of */ usr / local / tomcat / logs /*.
  - The log names match the form of *catalina .*. log* .

In this example, the `$ name` parameter in the environment variables indicates the catalina access indexes.

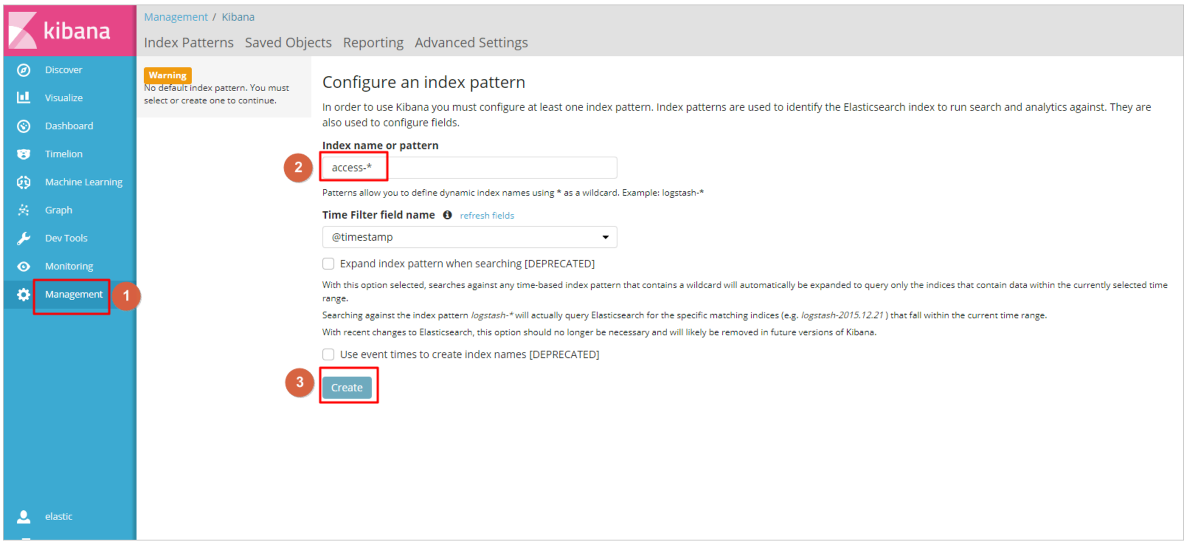
5. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Pods.

6. Select the target cluster and namespace where you deployed the Tomcat application. Then, in the Action column, choose More > Logs.



The following logs displayed in the Logs tab of the Pods - tomcat page indicates that the Tomcat application is deployed in the target Kubernetes cluster.

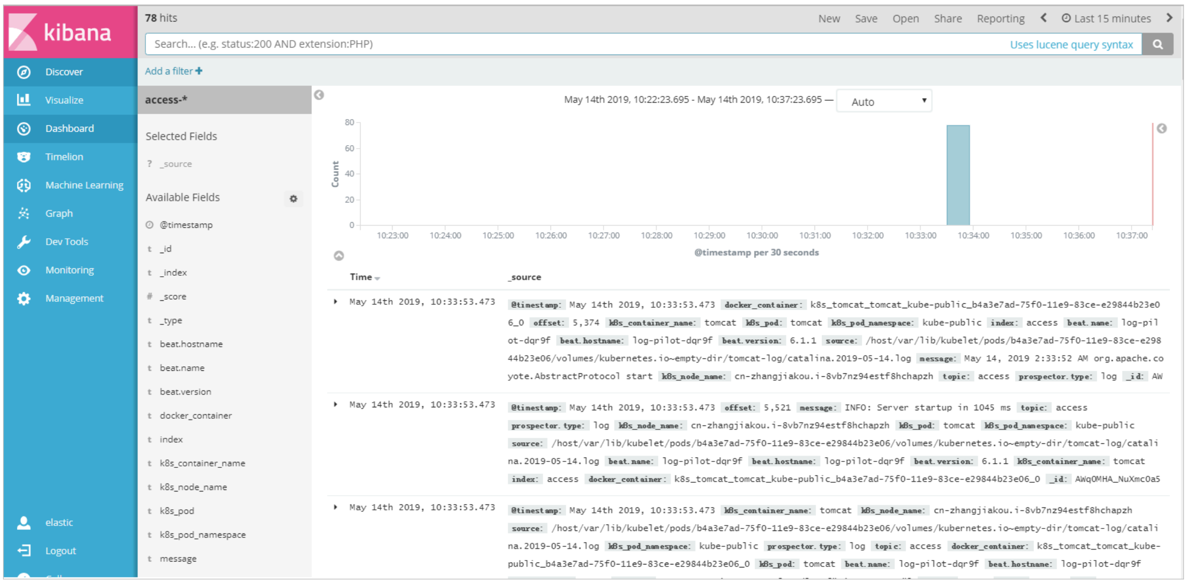
7. On the Elasticsearch console, click Kibana console to log on to Kibana.
8. In the left-side navigation pane, click Management. Then, in the Configure an index pattern area, enter `access-*`, and click Create.



9. (Optional): Create multiple indexes.

    a. Click Management. Then, in the Kibana area, click Index Patterns.
    b. Click Create Index Patterns. Then, in the `Index name or pattern` area, enter target indexes in the format of `XXX-*`, and click Create.

10.In the left-side navigation pane, click Discover to verify that the logs of the Tomcat application are collected to the target Elasticsearch instance.



> **Note:**
> By default, when Log-Pilot collects logs to an Elasticsearch instance, Log-Pilot creates indexes in the format of index-yyyy.MM.dd.

8.5 Configure Log4jAppender for Kubernetes and Log Service

Log4j is an open-source project of Apache, which consists of three important components: log level, log output destination, and log output format. By configurin g Log4jAppender, you can set the log output destination to console, file, GUI component, socket server, NT event recorder, or UNIX Syslog daemon.

This document introduces how to configure a YAML file to output Alibaba Cloud Container Service Kubernetes cluster logs to Alibaba Cloud Log Service, without modifying the application codes. In this document, deploy a sample API application in the Kubernetes cluster for demonstration.

Prerequisites

- You have activated Container Service and created a Kubernetes cluster.

  In this example, create a Kubernetes cluster in the region of China East 1 (Hangzhou).
- Enable AccessKey or Resource Access Management (RAM). Make sure you have sufficient access permissions. Use the AccessKey in this example.

Step 1 Configure Log4jAppender in Alibaba Cloud Log Service

1. Log on to the Log Service console.
2. On the Project List page, click Create Project in the upper-right corner. Complete the configurations and then click Confirm to create the project.

   In this example, create a project named k8s-log4j and select the same region (China East 1 (Hangzhou)) as the Kubernetes cluster.

   Note:
   Generally, create a Log Service project in the same region as the Kubernetes cluster. When the Kubernetes cluster and Log Service project are in the same region, log data is transmitted by using the intranet, which saves the Internet bandwidth cost and time of data transmission because of different regions, and implements the best practice of real-time collection and quick query.

3. After being created, the project k8s-log4j is displayed on the Project List page. Click the project name.
4. The Logstore List page appears. Click Create in the upper-right corner.
5. Complete the configurations and then click Confirm.

   In this example, create a Logstore named k8s-logstore.

6. Then, a dialog box asking you to use the data import wizard appears.
7. Click Data Import Wizard. In the Select Data Source step, select log4jAppender under Other Sources and then complete the configurations as instructed on the page.

   Use the default configurations in this example. Configure the settings according to the specific scenarios of log data.

Step 2 Configure Log4jAppender in the Kubernetes cluster

In this example, use the sample YAML files demo-deployment and demo-service for demonstration.

1. Connect to your Kubernetes cluster.

   For more information, see #unique_130 or #unique_19.

2. Obtain the *demo - deployment . yaml* file and configure the environment variable `JAVA_OPTS` to collect logs from the Kubernetes cluster.

   The sample orchestrat ion of the demo - deployment . yaml file is as follows:

```
apiVersion :  apps / v1beta2
kind :  Deployment
metadata :
  name :  log4j - appender - demo - spring - boot
  labels :
    app :  log4j - appender
spec :
  replicas :  1
  selector :
    matchLabel  s :
      app :  log4j - appender
  template :
    metadata :
    labels :
      app :  log4j - appender
  spec :
    containers :
   -  name :  log4j - appender - demo - spring - boot
      image :  registry . cn - hangzhou . aliyuncs . com /
jaegertrac  ing / log4j - appender - demo - spring - boot : 0 . 0 .
2
      env :
     -  name :  JAVA_OPTS  ## Note
        value : "- Dproject ={ your_proje  ct } - Dlogstore ={
your_logst  ore } - Dendpoint ={ your_endpo  int } - Daccess_ke
y_id ={ your_acces  s_key_id } - Daccess_ke  y ={ your_acces
s_key_secr  et }"
      ports :
     -  containerP  ort :  8080
```

   Wherein:

   - `- Dproject` : The name of the used Alibaba Cloud Log Service project. In this example, it is k8s-log4j.
   - `- Dlogstore` : The name of the used Alibaba Cloud Log Service Logstore. In this example, it is k8s-logstore.
   - `- Dendpoint` : The service endpoint of Log Service. You must configure your service endpoint according to the region where the Log Service project

resides. For more information, see Service endpoint. In this example, it is cn-hangzhou.log.aliyuncs.com.

- `- Daccess_ke y_id` : Your AccessKey ID.
- `- Daccess_ke y` : Your AccessKey Secret.

3. Run the following command in the command line to create the deployment:

```
kubectl  create  - f  demo - deployment . yaml
```

4. Obtain the *demo - service . yaml* file and run the following command to create the service.

   No need to modify the configurations in the *demo - service . yaml* file.

```
kubectl  create  - f  demo - service . yaml
```

Step 3 Test to generate Kubernetes cluster logs

You can run the `kubectl get` command to view the deployment status of the resource object. Wait until the deployment and the service are successfully deployed. Then, run the `kubectl get svc` command to view the external access IP of the service, that is, the EXTERNAL-IP.

```
$  kubectl  get  svc
 NAME  TYPE  CLUSTER - IP  EXTERNAL - IP  PORT ( S )  AGE
 log4j - appender - demo - spring - boot - svc  LoadBalanc er  172
 . 21 .  XX . XX  120 . 55 .  XXX . XXX  8080 : 30398 / TCP  1h
```

In this example, test to generate Kubernetes cluster logs by running the `login` command, wherein, `K8S_SERVIC E_IP` is the `EXTERNAL - IP` .

Note:

See GitHub log4j-appender-demo to view the complete collection of APIs.

```
curl  http ://${ K8S_SERVIC E_IP }: 8080 / login ?  name = bruce
```

Step 4 View logs in Alibaba Cloud Log Service

Log on to the Log Service console.

Click the project name and click Search at the right of the Logstore k8s-logstore to view the output logs of the Kubernetes cluster.

The output content of the log corresponds to the preceding command. This example demonstrates how to output the logs of the sample application to Alibaba Cloud Log Service. By completing the preceding steps, you can configure Log4JAppender in Alibaba Cloud and implement advanced functions such as collecting logs in real time, filtering data, and querying logs by using Alibaba Cloud Log Service.

9 Monitoring management

9.1 Deploy the Prometheus monitoring system

Prometheus is an open source monitoring tool for cloud native applications. This topic describes how to deploy the Prometheus monitoring system by using Alibaba Cloud Container Service for Kubernetes.

Prerequisites

- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have connected to the Master node so that you can view node labels and other information. For more information, see Connect to a Kubernetes cluster by using kubectl.

Context

A monitoring system monitors the following two types of objects:

- Resource, namely, the resource usage of a node or application. The monitoring system of Container Service for Kubernetes monitors node resource usage, cluster resource usage, and pod resource usage.
- Application, namely, internal metrics of an application. For example, The monitoring system collects statistics regarding the number of online users that use an application in real time, and performs service-level monitoring and alarming for the application by exposing ports.

The following are the objects monitored in a Kubernetes cluster:

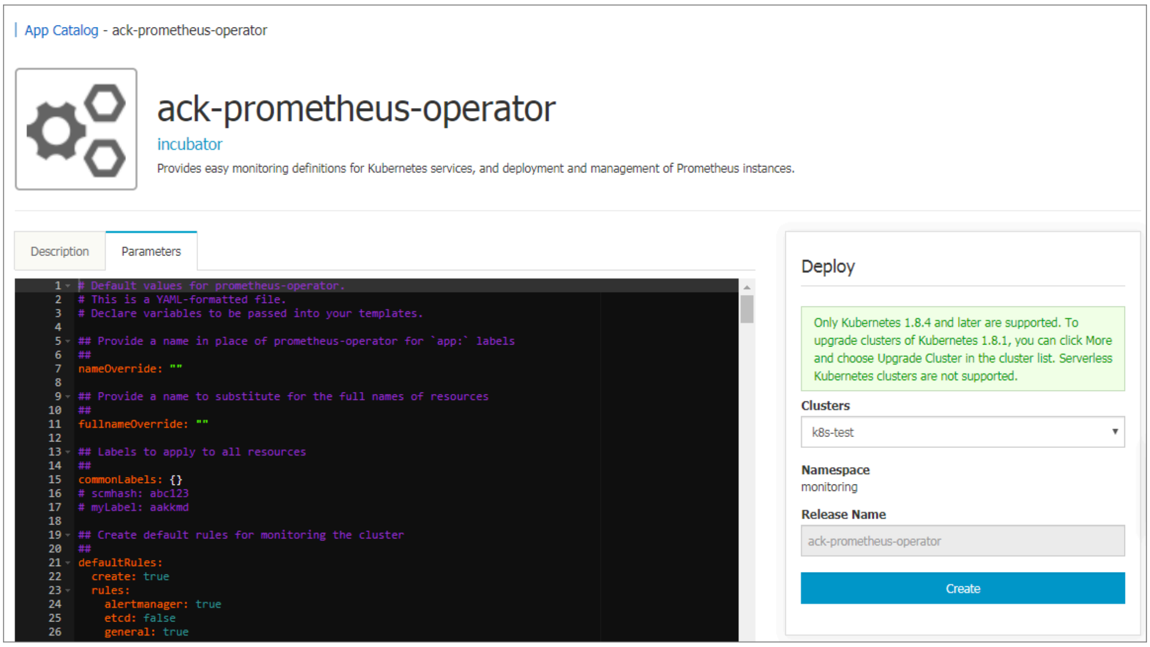
- System components, which are built-in components of the Kubernetes cluster, such as apiserver, controller-manager, and etcd.
- Static resource entities, which include node resource status and kernel events.
- Dynamic resource entities, which are abstract workload entities of Kubernetes, such as deployment, DaemonSet, and pods.
- Customized application objects, which includes the data and metrics that require customization within an application.

To monitor system components and static resource entities, you need to specify monitoring methods for them in the configuration file.

To monitor dynamic resource entities, we recommend that you deploy the Prometheus monitoring system.

Procedure

1. Deploy Prometheus monitoring system
   a) Log on to the Container Service console.
   b) In the left-side navigation pane under Container Service-Kubernetes, choose Marketplace > App Catalog. Then, click ack-prometheus-operator.
   c) In the Deploy area, select the target cluster. Then, click Create.



Verify the result

A. Run the following command to map the Prometheus that is deployed to the cluster to the local port 9090.

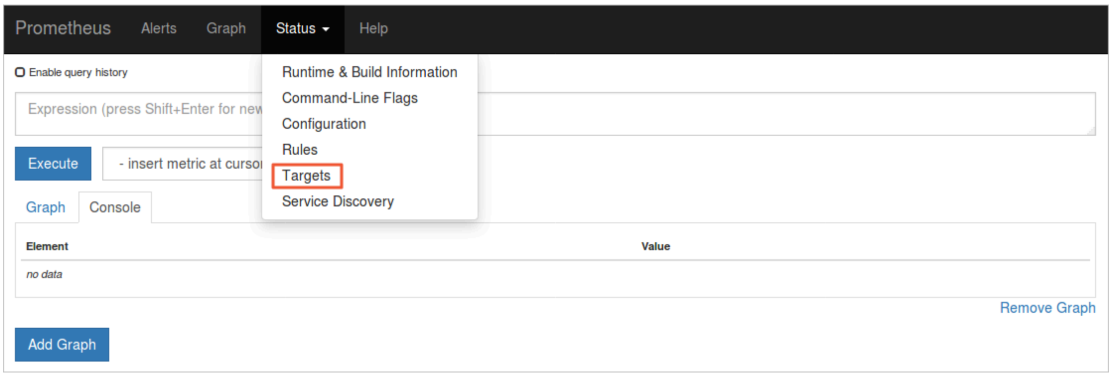
```
kubectl  port - forward  svc / ack - prometheus - operator -
prometheus  9090 : 9090  - n  monitoring
```

B. To view Prometheus, access `localhost : 9090` in a browser.

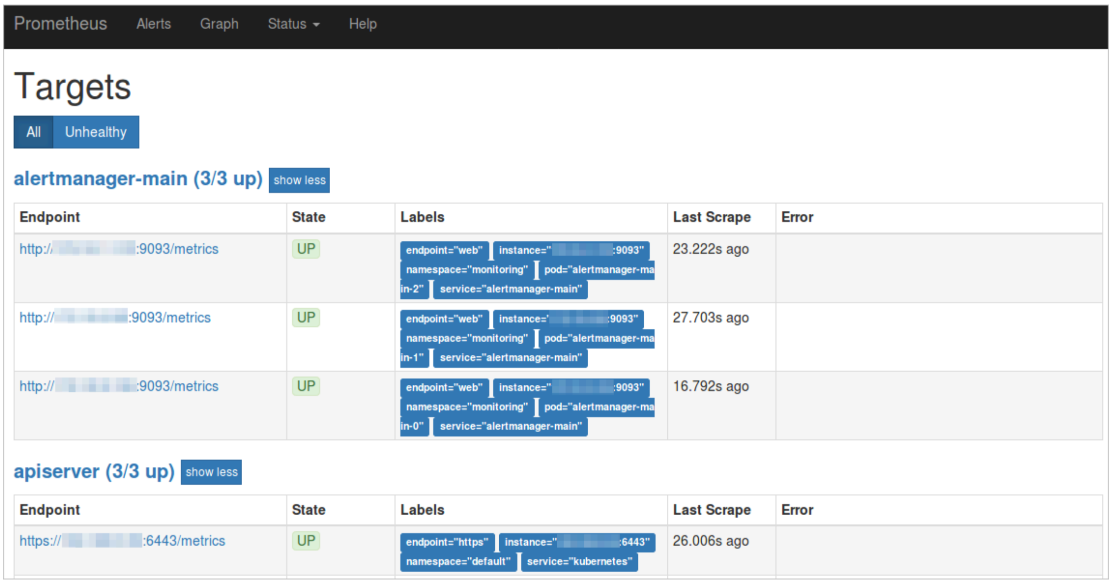
> **Note:**

By default, Prometheus cannot be accessed through the Internet. You must use your local proxy to access it.

C. Select Targets under the Status menu to view all collection tasks.



If the status of all tasks is UP, all collection tasks are running properly.



2. View and display data aggregation.

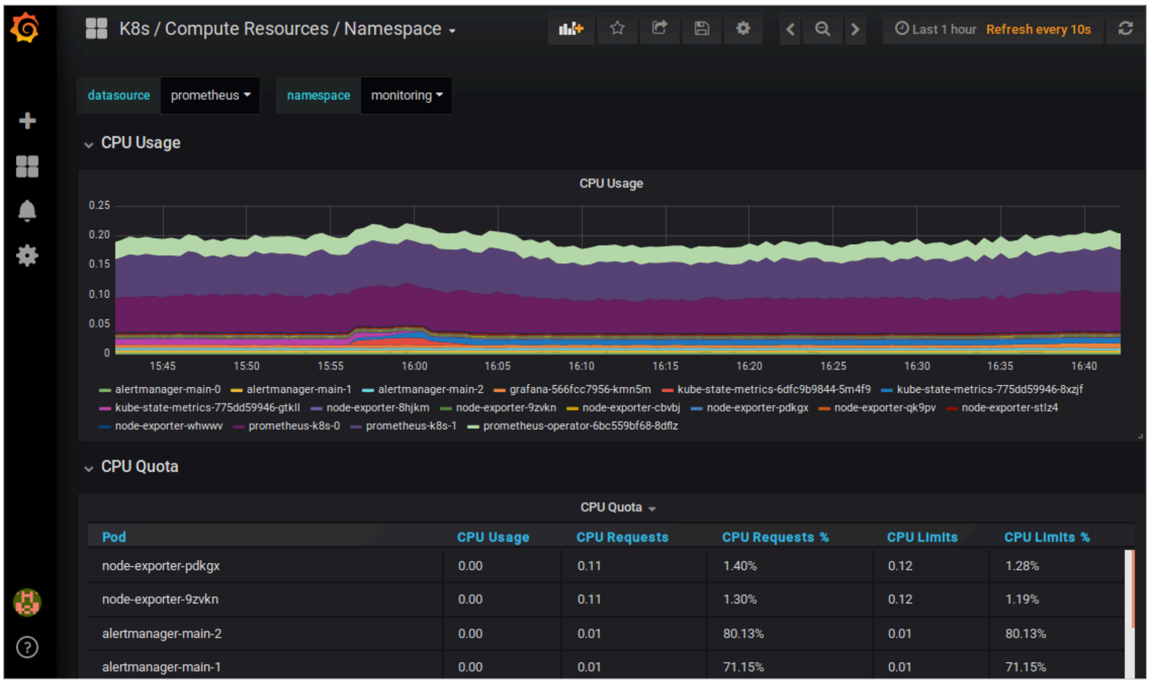
a) Run the following command to may the Grafana that is deployed to the cluster to the local port 3000.

```
kubectl  - n   monitoring   port - forward   svc / ack -
prometheus - operator - grafana   3000 : 80
```

b) Access `localhost : 3000` in your browser and then select a dashboard to view data aggregation.

Note:

The default user name and password are both admin.

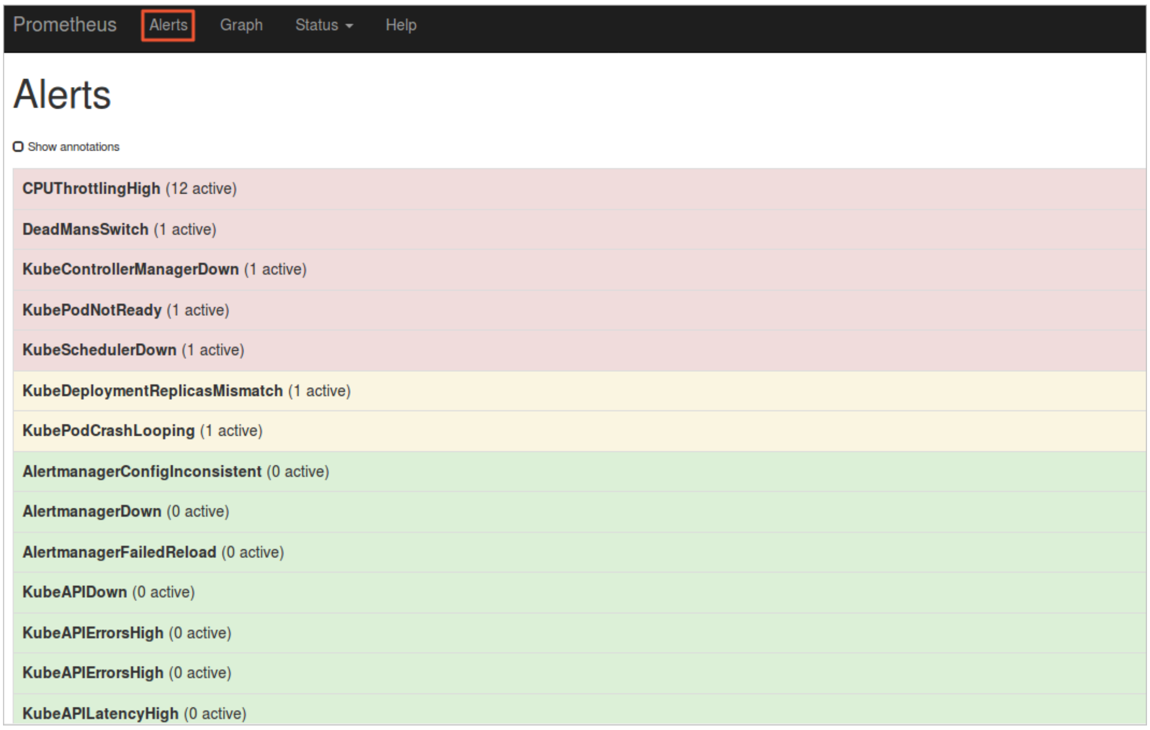


3. View alerting rules and set alert silencing.

- View alerting rules

  Access `localhost : 9090` in your browser and click the Alerts menu to view the current alerting rules.

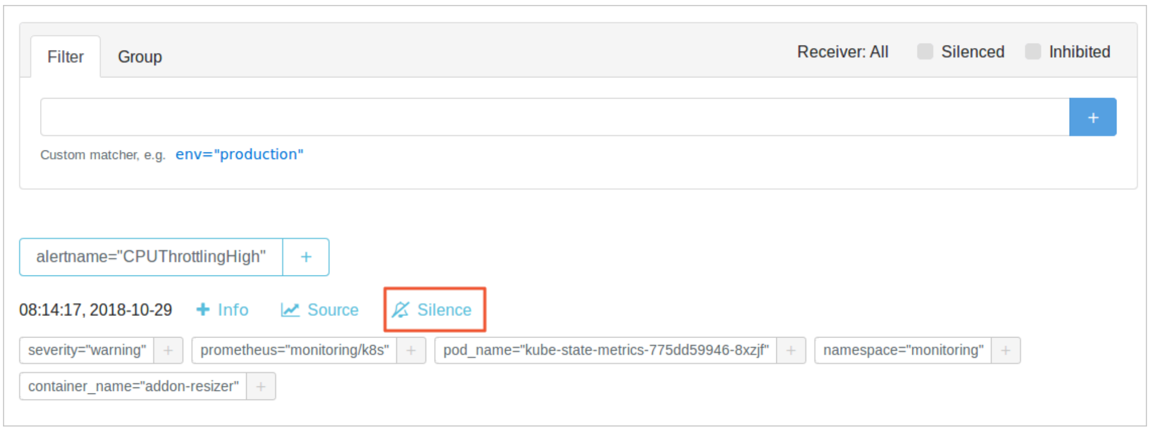
  - Red: indicates that an alert is triggered.
  - Green: indicates the normal status.



- Set alert silencing

  Run the following command, open `localhost : 9093` in your browser, and select Silenced to set alert silencing:

```
kubectl -- namespace monitoring port - forward svc /
alertmanag er - operated 9093
```



10 Security management

10.1 Security

Authorization

Kubernetes clusters support authorizing RAM users to perform operations on clusters.

For more information, see Use the Container Service console as a RAM user.

Full-link TLS certificates

The following communication links in Container Service Kubernetes clusters are verified by TLS certificates to prevent the communication from being eavesdropped or tampered:

- `kubelet` on worker nodes actively communicates with `apiserver` on master nodes
- `apiserver` on master nodes actively communicates with `kubelet` on worker nodes

During initialization, the master node uses SSH tunnels to connect to the SSH service of other nodes (port 22) for initialization.

Native secret & RBAC support

Kubernetes secrets are used to store sensitive information such as passwords, OAuth tokens, and SSH keys. Using plain text to write sensitive information to a pod YAML file or a Docker image may leak the information, while using secrets avoids such security risks effectively.

For more information, see Secret.

Role-Based Access Control (RBAC) uses the Kubernetes built-in API group to drive authorization and authentication, which allows you to use APIs to manage pods that correspond to different roles, and the access permissions of roles.

For more information, see Using RBAC authorization.

Network policy

In a Kubernetes cluster, pods on different nodes can communicate with each other by default. In some scenarios, to reduce risks, the network intercommunication among different business services is not allowed and you must introduce the network policy. In Kubernetes clusters, you can use the Canal network driver to implement the support for network policy.

Image security scan

Kubernetes clusters can use Container Registry to manage images, which allows you to perform image security scan.

Image security scan identifies the security risks in images quickly and reduces the possibility of applications running on your Kubernetes cluster being attacked.

For more information, see Image security scan.

Security group and Internet access

By default, each newly created Kubernetes cluster is assigned a new security group with the minimal security risk. This security group only allows ICMP for the Internet inbound.

By default, you cannot use Internet SSH to access your clusters. To use Internet SSH to connect to the cluster nodes, see Access Kubernetes clusters by using SSH.

The cluster nodes access the Internet by using the NAT Gateway, which further reduces the security risks.

10.2 Kube-apiserver audit logs

In a Kubernetes cluster, apiserver audit logs are important for cluster Operation & Maintenance (O&M) because they record daily operations of different users. This topic describes how to configure the apiserver audit logs of an Alibaba Cloud Kubernetes cluster, and how to collect and analyze audit logs through Log Service, and how to customize audit log alarm rules.

Configurations of apiserver audit logs

The apiserver audit function is enabled by default when you create a Kubernetes cluster. Relevant parameters and description are as follows:

Note:

Log on to the Master node, and the directory of the apiserver configuration files is */ etc / kubernetes / manifests / kube - apiserver . yaml .*

Configuration	Description
audit-log-maxbackup	The maximum fragment of audit logs stores 10 log files.
audit-log-maxsize	The maximum size of a single audit log is 100 MB.
audit-log-path	The audit log output path is */ var / log / kubernetes / kubernetes . audit .*
audit-log-maxage	The longest storage period of audit logs is seven days.
audit-policy-file	Configuration policy file of audit logs. The directory is */ etc / kubernetes / audit - policy . yml .*

Log on to the Master node machine. The directory of the audit log configuration policy file is */ etc / kubernetes / audit - policy . yml* . The content of the file is as follows:

```
apiVersion : audit . k8s . io / v1beta1 # This is required .
kind : Policy
# We recommend that you do not generate audit events
  for all requests in RequestRec eived stage .
omitStages :
 - " RequestRec eived "
rules :
 # The following requests are manually identified as
high - volume and low - risk .
 # Therefore , we recommend that you 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 : serviceacc  ount : 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 "]
 #  We   recommend   that   you   do   not   log   these   read -
only   URLs .
 -  level :  None
    nonResourc  eURLs :
     - / healthz *
     - / version
     - / swagger *
 #  We   recommend   that   you   do   not   log   events   requests
.
 -  level :  None
    resources :
     -  group : "" #  core
        resources : [" events "]
 #  Secrets ,  ConfigMaps ,  and   TokenRevie  ws   can   contain
sensitive   and   binary   data .
 #  Therefore ,  they   are   logged   only   at   the   Metadata
level .
 -  level :  Metadata
    resources :
     -  group : "" #  core
        resources : [" secrets ", " configmaps "]
     -  group :  authentica  tion . k8s . io
        resources : [" tokenrevie  ws "]
 #  Get   repsonses   can   be   large ;  skip   them .
 -  level :  Request
    verbs : [" get ", " list ", " watch "]
    resources :
     -  group : "" #  core
     -  group : " admissionr  egistratio  n . k8s . io "
     -  group : " apps "
     -  group : " authentica  tion . k8s . io "
     -  group : " authorizat  ion . k8s . io "
     -  group : " autoscalin  g "
     -  group : " batch "
     -  group : " certificat  es . k8s . io "
     -  group : " extensions "
     -  group : " networking . k8s . io "
     -  group : " policy "
```

```
      - group : " rbac . authorizat  ion . k8s . io "
      - group : " settings . k8s . io "
      - group : " storage . k8s . io "
  # Default  level  for  known  APIs .
  - level : RequestRes ponse
    resources :
      - group : "" # core
      - group : " admissionr egistratio n . k8s . io "
      - group : " apps "
      - group : " authentica tion . k8s . io "
      - group : " authorizat ion . k8s . io "
      - group : " autoscalin g "
      - group : " batch "
      - group : " certificat es . k8s . io "
      - group : " extensions "
      - group : " networking . k8s . io "
      - group : " policy "
      - group : " rbac . authorizat  ion . k8s . io "
      - group : " settings . k8s . io "
      - group : " storage . k8s . io "
  # Default  level  for  all  other  requests .
  - level : Metadata
```

Note:

- Logs are not recorded immediately after requests are received. Log recording starts only after the response body header is sent.
- The following requests or operations are not audited: redundant kube-proxy watch requests, GET requests from kubelet and system:nodes for nodes, operations performed on endpoints by kube components in the kube-system, and GET requests from the apiserver for namespaces.
- Read-only urls such as `/ healthz *`, `/ version *`, and `/ swagger *` are not audited.
- Logs of interfaces of secrets, configmaps, and tokenreviews are set to the metadata level because they might contain sensitive information or binary files . For logs of this level, only the user, timestamp, request resources, and request actions of the request event are audited. The request body and the response body are not audited.
- For sensitive interfaces such as authentication, rbac, certificates, autoscaling, and storage, the corresponding request bodies and response bodies are audited according to the read and write requests.

View audit log reports

A Kubernetes cluster that runs on Alibaba Cloud Container Service has three audit log reports that provide the following information:

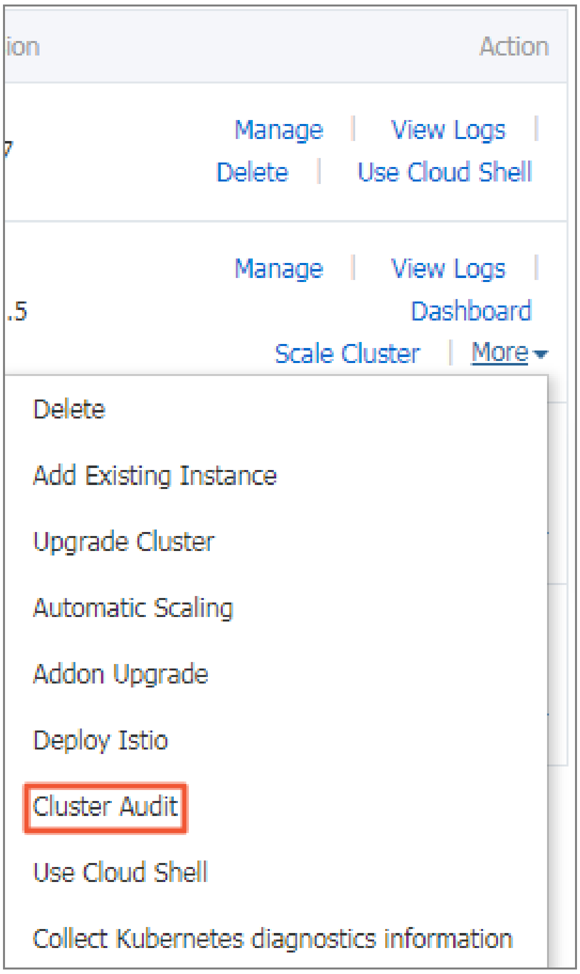
- Operations performed by all users and system components on the cluster
- The source IP address of each operation, the area to which a source IP addresses belongs, and the source IP address distribution
- Detailed operation charts of all resources
- Operation charts of each sub-account
- Charts of important operations such as logging on to a container, accessing a secret, and removing resources

Note:

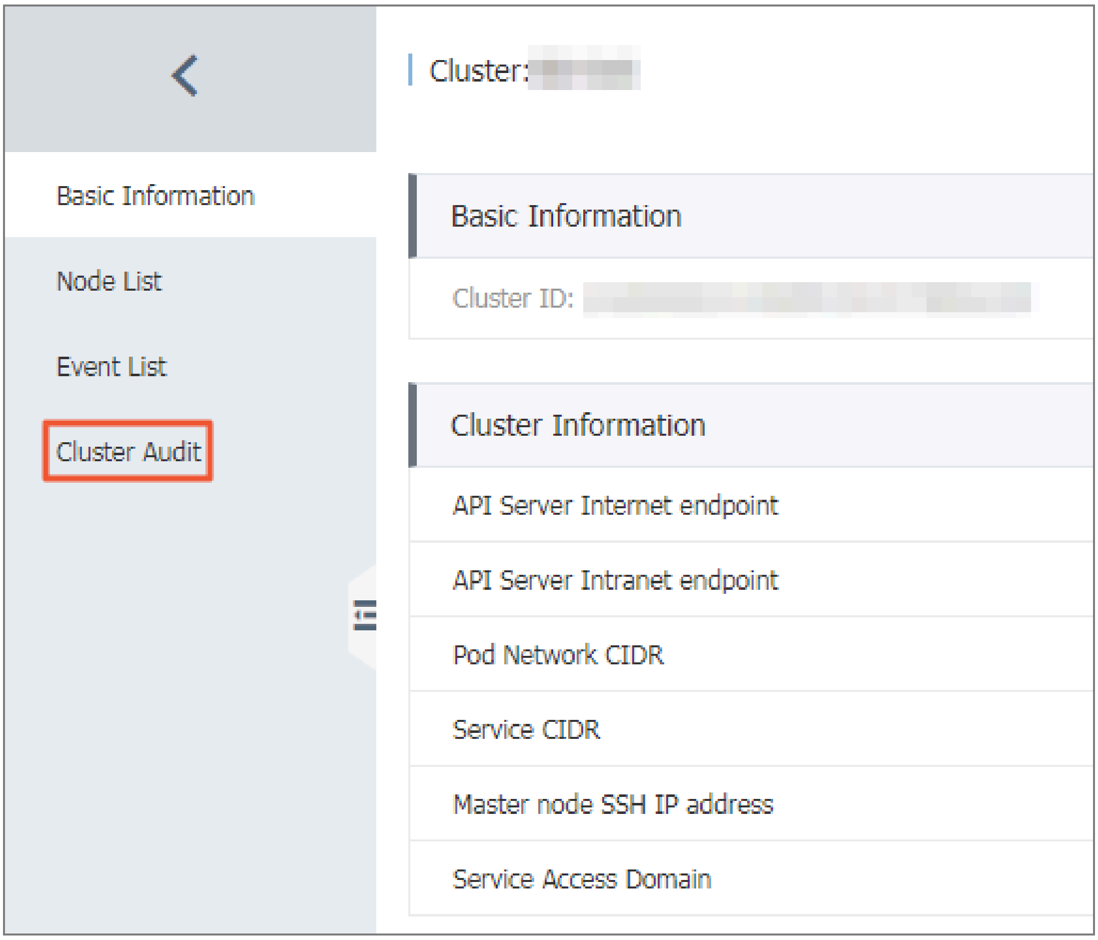
- For Kubernetes clusters created after January 13, 2019, if you active Log Service for the clusters, the system automatically enables audit log report functions. If audit log report functions are disabled for a Kubernetes cluster, see Manually enable audit log report functions.
- We recommend that you do not modify audit log reports. If you want to customize audit log reports, you can create new reports in the Log Service console.

You can access audit log reports by using either of the following two methods:

- Log on to the Container Service console. In the action column of the target cluster, chooseMore > Cluster Audit.



- Log on to the Container Service console. Click the target cluster name, and then click Cluster Auditin the left-side navigation pane.

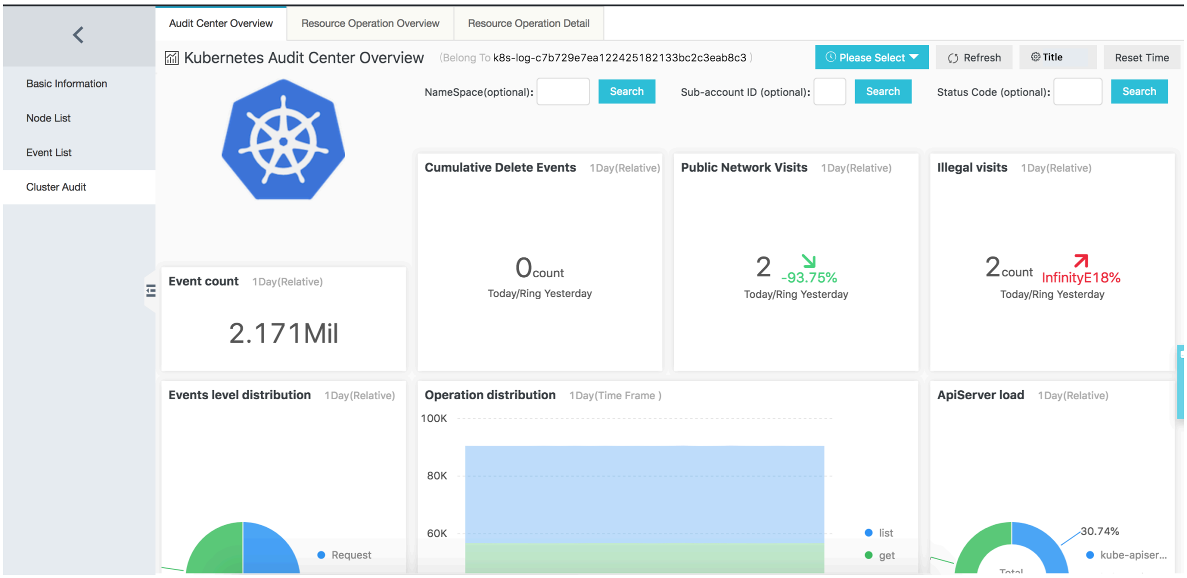


Audit log report overview

The following three apiserver audit log reports are available: Audit Center Overview, Resource Operation Overview, and Resource Operation Detail.

- Audit Center Overview

This report displays an overview of the Kubernetes cluster events and the detailed information about important events, such as public network visits, command execution, resource removal, and secret visits.



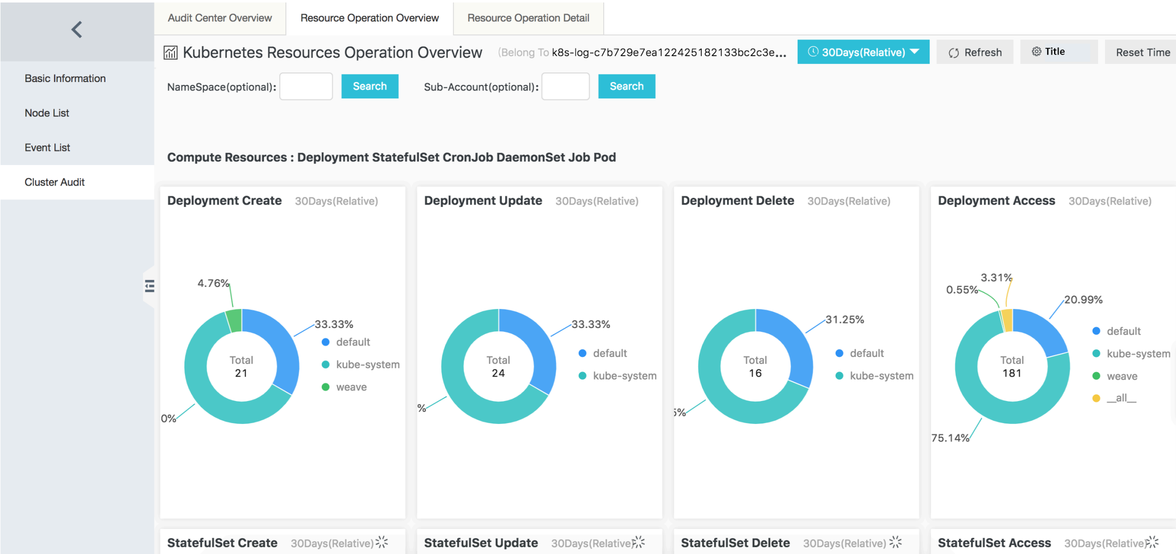
Note:

By default, this report displays statistics for one week. You can customize a statistics time range. In addition, you can filter events by specifying one or multiple factors, such as a namespace, a sub-account ID, and a status code.

- Resource Operation Overview

  This report displays the operation statistics information about computing resources, network resources, and storage resources of a Kubernetes cluster. Operations include creation, update, removal, and access.

  - 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.

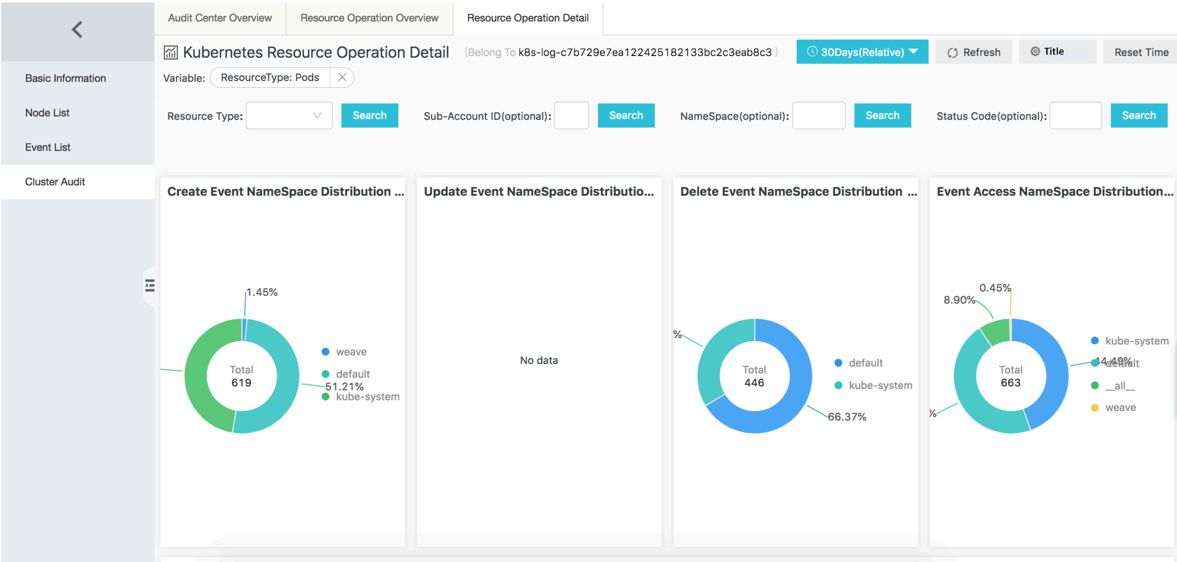


Note:

- By default, this report displays statistics for one week. You can customize a statistics time range. In addition, you can filter events by specifying one or both of the following factors: a namespace or a sub-account ID.
- If you want to view the detailed operation events of a resource, we recommend that you use Resource Operation Detail.

- Resource Operation Detail

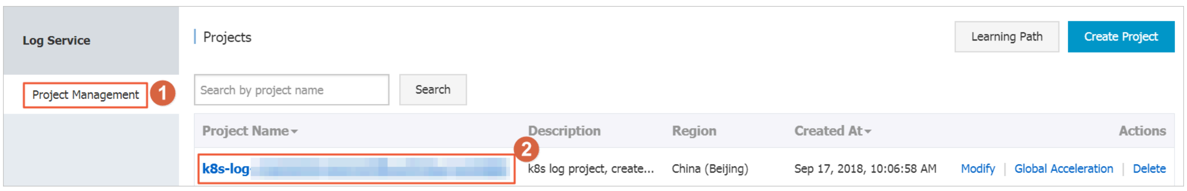
  This report displays detailed operation information of a Kubernetes cluster resource. You must select or enter a resource type to view detailed operation information in time. This report displays the total number of operation events, namespace distribution, success rate, timing trend, and specific operation charts.



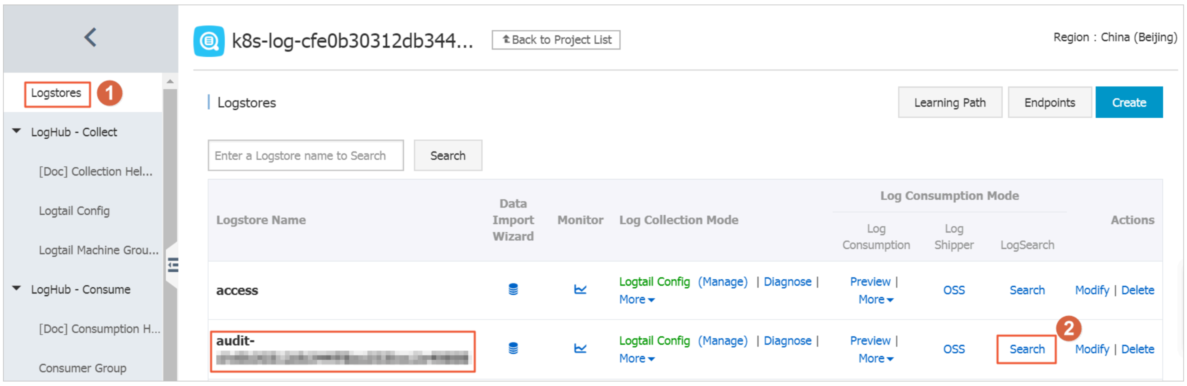
> **Note:**
>
> - If you want to view a CRD resource registered in Kubernetes or any other resources not listed in the report, you can enter the plural form of the target resource. For example, to view a CRD resource named AliyunLogConfig, you can enter AliyunLogConfigs.
> - By default, this report displays statistics for one week. You can customize a statistics time range. In addition, you can filter events by specifying one or multiple factors, such as a namespace, a sub-account ID, and a status code.

logs, you can log on to Log Service to view detailed log records.

1. Log on to the Log Service console.
2. In the left-side navigation pane, click Project Management, select the Project configured when you create the cluster, and then click the Project name.



3. On the Logstores page, find the Logstore named audit-${clusterid} and click Search at the right side of the Logstore. The audit logs of the cluster are stored in this Logstore.



> **Note:**
> - When you create a Kubernetes cluster, your specified log Project automatically creates a Logstore named audit-${clusterid}.
> - The audit log Logstore index is set by default. We recommend that you do not modify the index. Otherwise, the audit log reports become invalid.

To searchfor an audit log, you can use one of the following methods:

- To querya sub-account operation record, enter the sub-account ID and then click Search & Analysis.
- To query operations on a resource, enter the resource name and click Search & Analysis.
- To filter out 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 then click Search & Analysis.

For more information, see Log Service search and analysis methods.

Set resource alarms

You can use the alarm function of Log Service to set resource alarms. Alarm notifications can be sent through a DingTalk group robot, a customized Webhook, and the Message Center.

Note:

Audit log reports provide multiple query statements. On the Logstores page, click Dashboardin the left-side navigation pane and then click a dashboard (namely, an audit log report) to display all charts, Click the menu in the upper-right corner of a chart, and then click View Details.

Example 1: Set an alarm notification for running a command on a container

To prevent Kubernetes cluster users from logging on to any container to run a command, you must set an alarm notification for running a command on any container. Furthermore, the alarm notification must include detailed information such as the container to which the user logged on, commands, user name, the event ID, the operation time, and the user IP address.

- The query statement is as follows:

```
verb  :  create   and   objectRef . subresourc  e : exec   and
stage :   ResponseSt  arted  |  SELECT   auditID   as  " event   ID
",  date_forma  t ( from_unixt  ime ( __time__ ), '% Y -% m -% d
% T ' )  as  " operation   time ",   regexp_ext  ract (" requestURI
", '([^\?]*)/ exec \?.*',  1 ) as  " resource ",   regexp_ext  ract
(" requestURI ", '\?(.*)',  1 ) as  " command " ," responseSt  atus
. code "  as  " status   code ",
 CASE
 WHEN  " user . username " != ' kubernetes - admin '  then  " user
. username "
 WHEN  " user . username " = ' kubernetes - admin '  and
regexp_lik  e (" annotation  s . authorizat  ion . k8s . io
/ reason ", ' RoleBindin  g ')  then   regexp_ext  ract ("
annotation  s . authorizat  ion . k8s . io / reason ", '  to   User
 "(\ w +)"',  1 )
 ELSE  ' kubernetes - admin '  END
 as  " operation   account ",
CASE   WHEN   json_array  _length ( sourceIPs ) =  1   then
json_forma  t ( json_array  _get ( sourceIPs ,  0 ))  ELSE
sourceIPs   END
as  " source   IP   address "  limit   100
```

- The condition expression is `operation   event  =~ ".*"`.

Example 2: Set an alarm notification for failed Internet access to apiserver

To prevent malicious attacks on a Kubernetes cluster for which Internet access is enabled, you need to monitor the number of Internet access times and the failed access rate. Specifically, an alarm notification must be sent, when the number of Internet access times reaches a specified threshold and the failed access rate exceeds a specified threshold. Furthermore, the alarm notification must include detailed information such asto which the user IP address belongs, the user IP address, and the high risk IP address. For example, to receive an alarm notification when the number of Internet access times reaches 10 and the failed access rate exceeds 50%, configure the following settings:

- Query statement.

```
* | select ip as " source IP address ", total as "
 number of access times ", round ( rate * 100 , 2 ) as
  " failed access rate %", failCount as " number of
 illegal access times ", CASE when security_c heck_ip
 ( ip ) = 1 then ' yes ' else ' no ' end as " high
 risk IP address ", ip_to_coun try ( ip ) as " country ",
 ip_to_prov ince ( ip ) as " province ", ip_to_city ( ip ) as
  " city ", ip_to_prov ider ( ip ) as " network operator "
 from ( select CASE WHEN json_array _length ( sourceIPs )
 = 1 then json_forma t ( json_array _get ( sourceIPs , 0 ))
 ELSE sourceIPs END
 as ip , count ( 1 ) as total ,
 sum ( CASE WHEN " responseSt atus . code " < 400 then 0
 ELSE 1 END ) * 1 . 0 / count ( 1 ) as rate ,
 count_if (" responseSt atus . code " = 403 ) as failCount
 from log group by ip limit 10000 ) where
 ip_to_doma in ( ip ) != ' intranet ' having " number of
 access times " > 10 and " failed access rate %" > 50
 ORDER by " number of access times " desc limit 100
```

- Condition expression is `source IP address =~ ".*"`.

Manually enable audit log report functions

You can manually enable audit log report functions.

1. Enable API server audit log.

   View the API server pod settings of the three Master nodes. That is, check whether audit log settings are configured for the startup parameters, the policy file, the environment variable, and the mounting directory.

   - Startup parameters

```
containers :
 - command :
   - kube - apiserver
   - -- audit - log - maxbackup = 10
   - -- audit - log - maxsize = 100
   - -- audit - log - path =/ var / log / kubernetes / kubernetes
. audit
   - -- audit - log - maxage = 7
```

```
    - -- audit - policy - file =/ etc / kubernetes / audit -
policy . yml
```

- Policy file (stored in the */ etc / kubernetes / audit - policy . yml* directory)

  For more information, see Configure a policy file.

  > **Note:**
  > If the */ etc / kubernetes /* directory does not have any policy file, you need to run the `vi audit - policy . yml` command to create a file, and then copy the content of the policy file and paste the content to the created file.

- Environment variable

```
env :
    - name :  aliyun_log  s_audit - c12ba20 *************** 9f0b
      value : / var / log / kubernetes / kubernetes . audit
    - name :  aliyun_log  s_audit - c12ba20 ***************
9f0b_tags
      value :  audit = apiserver
    - name :  aliyun_log  s_audit - c12ba20 ***************
9f0b_produ  ct
      value :  k8s - audit
    - name :  aliyun_log  s_audit - c12ba20 ***************
9f0b_jsonf  ile
      value : " true "
```

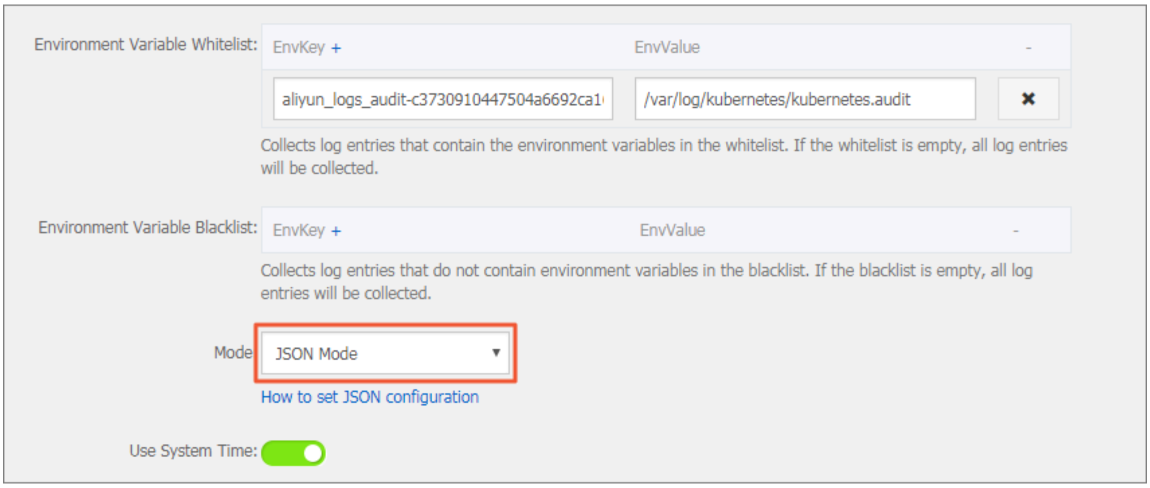
- Mounting directory

```
volumeMoun  ts :
    - mountPath : / var / log / kubernetes
      name :  k8s - audit
    - mountPath : / etc / kubernetes / audit - policy . yml
      name :  audit - policy
      readOnly :  true
volumes :
    - hostPath :
      path : / var / log / kubernetes
      type :  DirectoryO  rCreate
      name :  k8s - audit
    - hostPath :
      path : / etc / kubernetes / audit - policy . yml
      type :  FileOrCrea  te
      name :  audit - policy
```

Backup the original YAML file and then restart the API server by using a new *kube - apiserver . yaml* YAML file. This action will overwrite the original YAML file stored in the / etc / kubernetes / manifests / kube - apiserver . yaml directory.

If the API server pod settings does not contain the preceding settings, you must upgrade the Kubernete cluster to the latest version. For more information, see Upgrade a Kubernetes cluster.

2. Use the latest version of the Log Service component.
   - For how to install the Log Service component, see Manually install the Log Service component.
   - If you have installed the Log Service component, but the audit log function is disabled, you must upgrade the component to the latest version and you must ensure that your Logtail version is not earlier than v0.16.16 and can run on Master nodes. For more information, see Upgrade the Log Service component.
3. Update audit log parsing methods.
   a. Log on to the Log Service console.
   b. In the left-side navigation pane, click Project Management, and then click the name of the Project specified when creating your Kubernetes cluster.
   c. The Logstores page is displayed by default. Click Manage on the right of the Logstore named audit-${clustered}, and then click the configuration name. On the Specify Collection Mode tab page, select the JSON Mode.



Use a thirty-party log solution

Log on to the Master node of the cluster, and you can find the source file of the audit logs in the path of / var / log / kubernetes / kubernetes . audit . The source file is in standard JSON format. When deploying a cluster, you can use other log solutions to collect and search audit logs, instead of using Alibaba Cloud Log Service.

10.3 Implement secure access through HTTPS in Kubernetes

A Container Service Kubernetes cluster supports multiple application access methods. The most common methods include `SLB : Port` access, `NodeIP : NodePort` access, and domain name access. By default, a Kubernetes cluster does not support HTTPS access. To access applications through HTTPS, you can use the secure HTTPS access method provided by Container Service and Alibaba Cloud Server Load Balancer (SLB) service. This document explains how to configure a certificate in Container Service Kubernetes by using HTTPS access configuration as an example.

Depending on different access methods, your certificate can be configured with the following two methods:

- Configure the certificate on the frontend SLB.
- Configure the certificate on Ingress.

Prerequisites

- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have connected to the Master node through SSH. For more information, see Access Kubernetes clusters by using SSH.
- After connecting to the Master node, you have created the server certificates for the cluster, including the public key certificate and the private key certificate by running the following commands :

```
$   openssl   genrsa   - out   tls . key   2048

 Generating   RSA   private   key ,  2048   bit   long   modulus
................................................................ +++
.....................................................................................
+++
 e   is   65537  ( 0x10001 )

$   openssl   req  - sha256  - new  - x509  - days   365  - key
 tls . key  - out   tls . crt

 You   are   about   to   be   asked   to   enter   informatio  n
 that   will   be   incorporat  ed
...
-----
 Country   Name  ( 2   letter   code ) [ XX ]: CN
 State   or   Province   Name  ( full   name ) []: zhejiang
 Locality   Name  ( eg ,  city ) [ Default   City ]: hangzhou
 Organizati  on   Name  ( eg ,  company ) [ Default   Company   Ltd
 ]: alibaba
 Organizati  onal   Unit   Name  ( eg ,  section ) []: test
```

```
Common Name ( eg , your name or your server ' s
hostname ) []: foo . bar . com # you must configure
  the domain name correctly
Email Address []: a @ alibaba . com
```

Method 1: Configure the HTTPS certificate on SLB

This method has the following advantages and disadvantages:

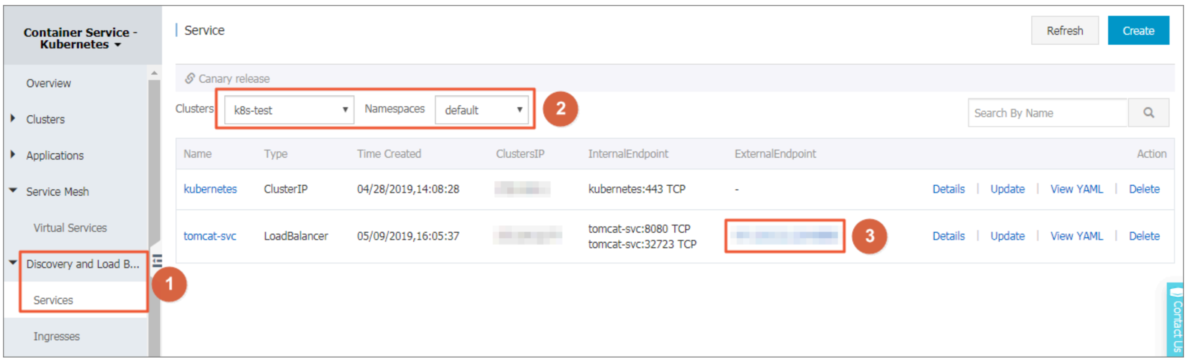
- Advantages: The certificate is configured on SLB and it is the external access portal of applications. The access to applications in the cluster still uses the HTTP access method.
- Disadvantages: You need to maintain many associations between domain names and their corresponding IP addresses.
- Scenarios: This method is applicable to applications that use LoadBalancer service rather than Ingress to expose access methods.

Preparations

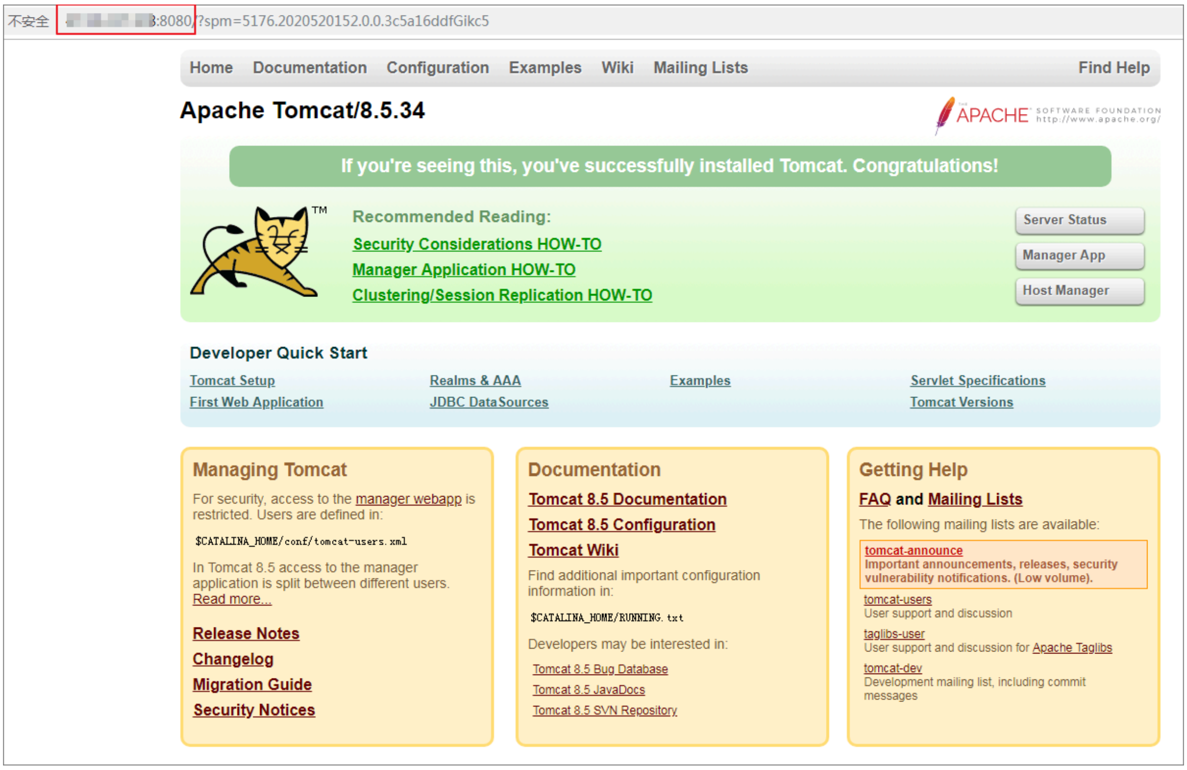
You have created a Tomcat application in the Kubernetes cluster. The application provides external access by using the LoadBalancer service. For more information, see Create a service.

Example

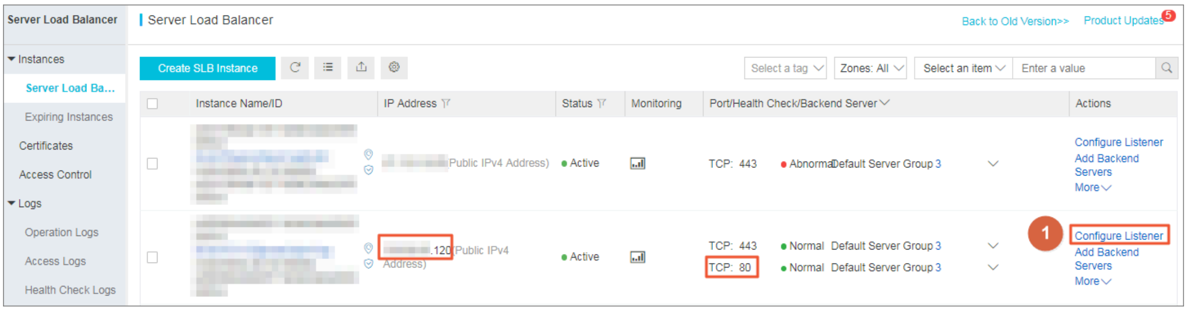
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Discovery and Load Balancing > Services. Then, select the cluster and the namespace to view the pre-created Tomcat application. As shown in the following figure, the created Tomcat application is named tomcat and the service name is tomcat-svc. The service type of the application is LoadBalancer, and the service port exposed by the application is 8080.



3. By clicking the external endpoint, you can access the Tomcat application through `IP : Port`.



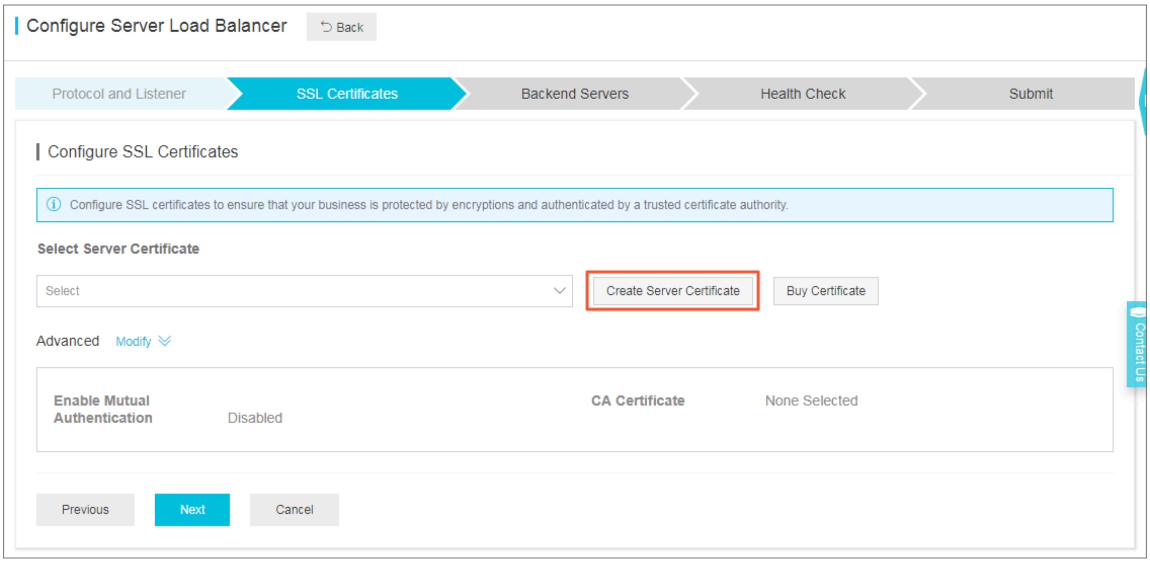
4. Log on to the SLB console.
5. By default, the Server Load Balancer page is displayed. In the IP address column, find the server load balancer that corresponds to the external endpoint of the tomcat-svc service, and click Configure Listener in the actions column.



6. Configure the server load balancer. Select a listener protocol first. Select HTTPS, set the listening port to 443, and then click Next.

7. Configure the SSL certificate.

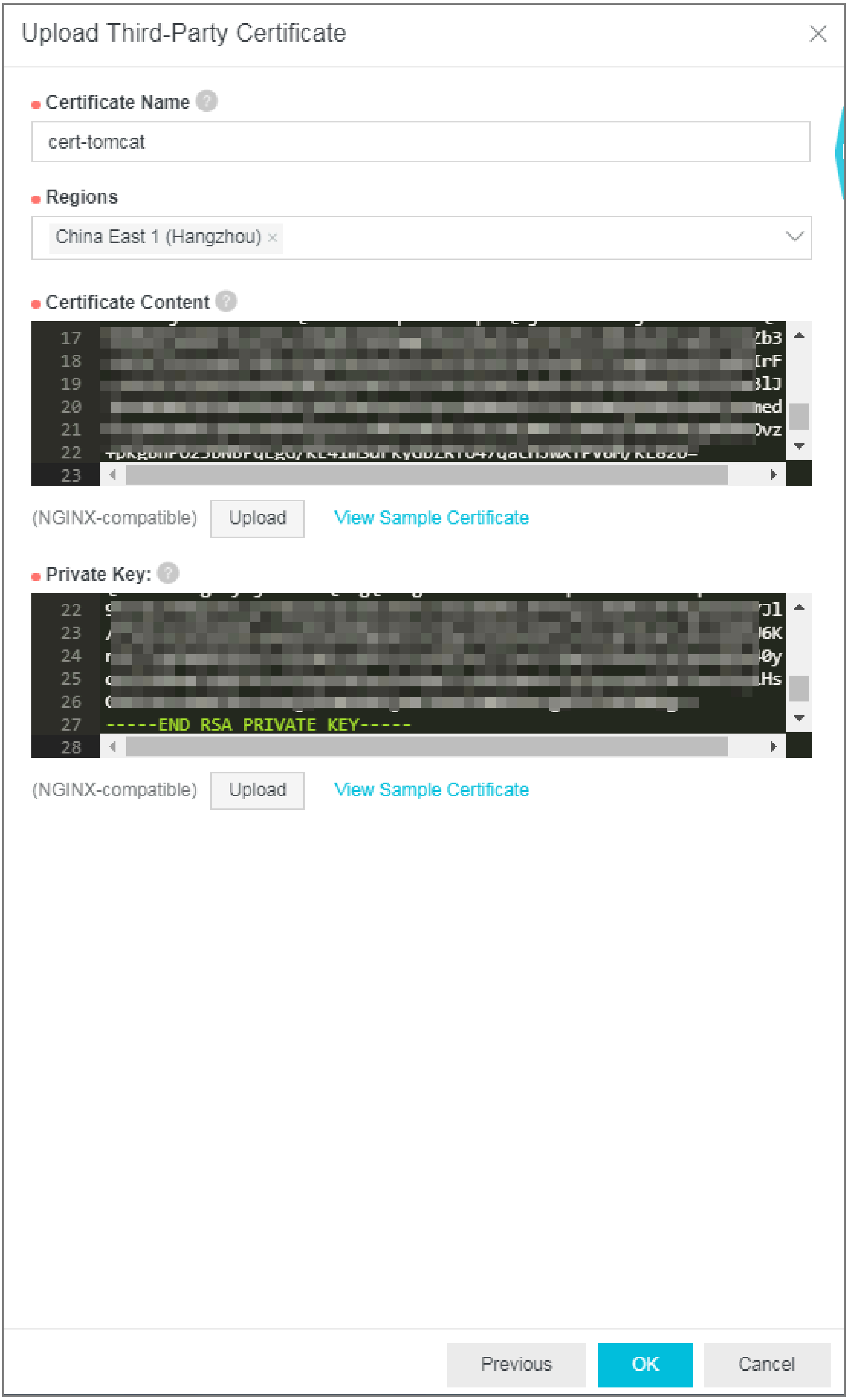
a. Click Create Server Certificate.



b. On the displayed page, select a certificate source. In this example, select Upload Third-Party Certificate, and then click Next.

c. On the uploading third-party certificate page, set the certificate name and select the region in which the certificate is deployed. In the Certificate Content and

the Private Key columns, enter the server public key certificate and private key created in Prerequisites, and then click OK.

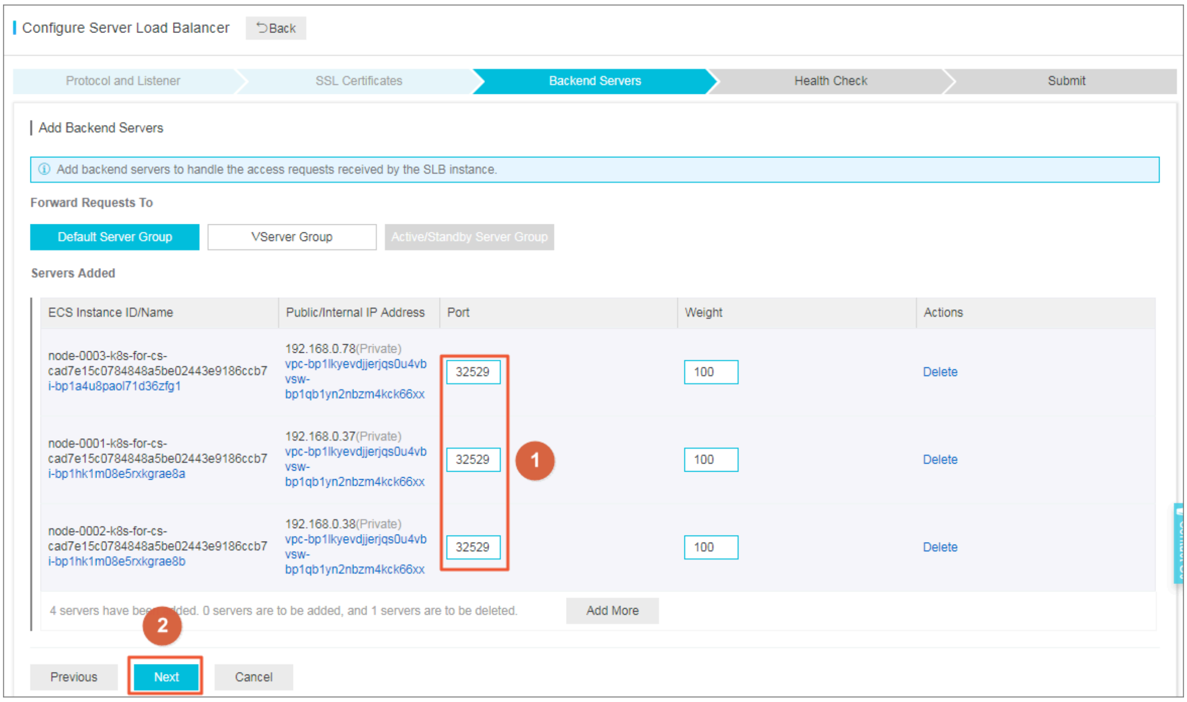

Upload Third-Party Certificate
Certificate Name
cert-tomcat
Regions
China East 1 (Hangzhou)
Certificate Content
(NGINX-compatible)
Upload
View Sample Certificate
Private Key:
-----END RSA PRIVATE KEY-----
(NGINX-compatible)
Upload
View Sample Certificate
Previous
OK
Cancel

d. From the Select Server Certificate drop-down list, select the created server certificate.

e. Click Next.

8. Configure Backend Servers. By default, servers are added. You need to configure a port for each backend server to listen to the tomcat-svc service, and then click Next.

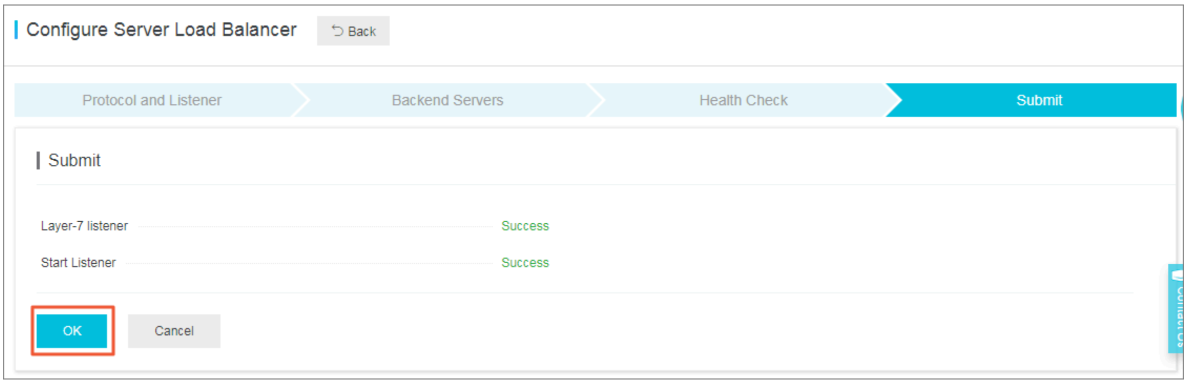
> **Note:**
> You need to find the NodePort number of this service in the Container Service Web interface, and configure the number as the port number of each backend server.



9. Configure Health Check, and then click Next. In this example, use the default settings.

10. Confirm the Submit tab. When you make sure that all configurations are correct, click Submit.

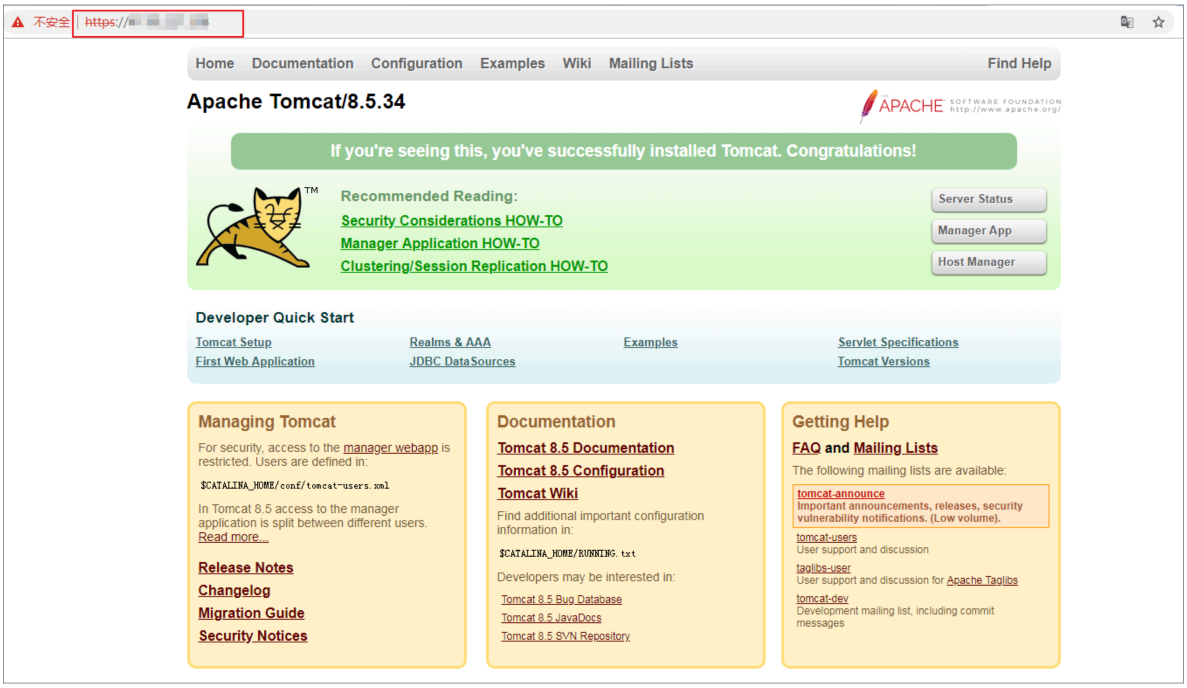
11.After completing the configuration, click OK.



12.Return to the Server Load Balancer page to view the instance. The listening rule of `HTTPS : 443` is generated.

13.Access the Tomcat application through HTTPS. In the address bar of the browser, enter `https :// slb_ip` to access the application.

> **Note:**
> If the domain name authentication is included in the certificate, you can access the application by using the domain name. You can also access the application through `slb_ip : 8080` because `tcp : 8080` is not deleted.



Method 2: Configure the certificate on Ingress

This method has the following advantages and disadvantages:

- Advantages: You do not need to modify the SLB configuration. All applications can manage their own certificates through Ingress without interfering with each other.
- Disadvantages: Each application can be accessed by using a separate certificate or the cluster has applications that can be accessed by only using a certificate.

Preparations

You have created a Tomcat application in the Kubernetes cluster. The service of the application provides access through ClusterIP. In this example, use Ingress to provide the HTTPS access service.

Example

1. Log on to the Master node of the Kubernetes cluster and create a secret according to the prepared certificate.

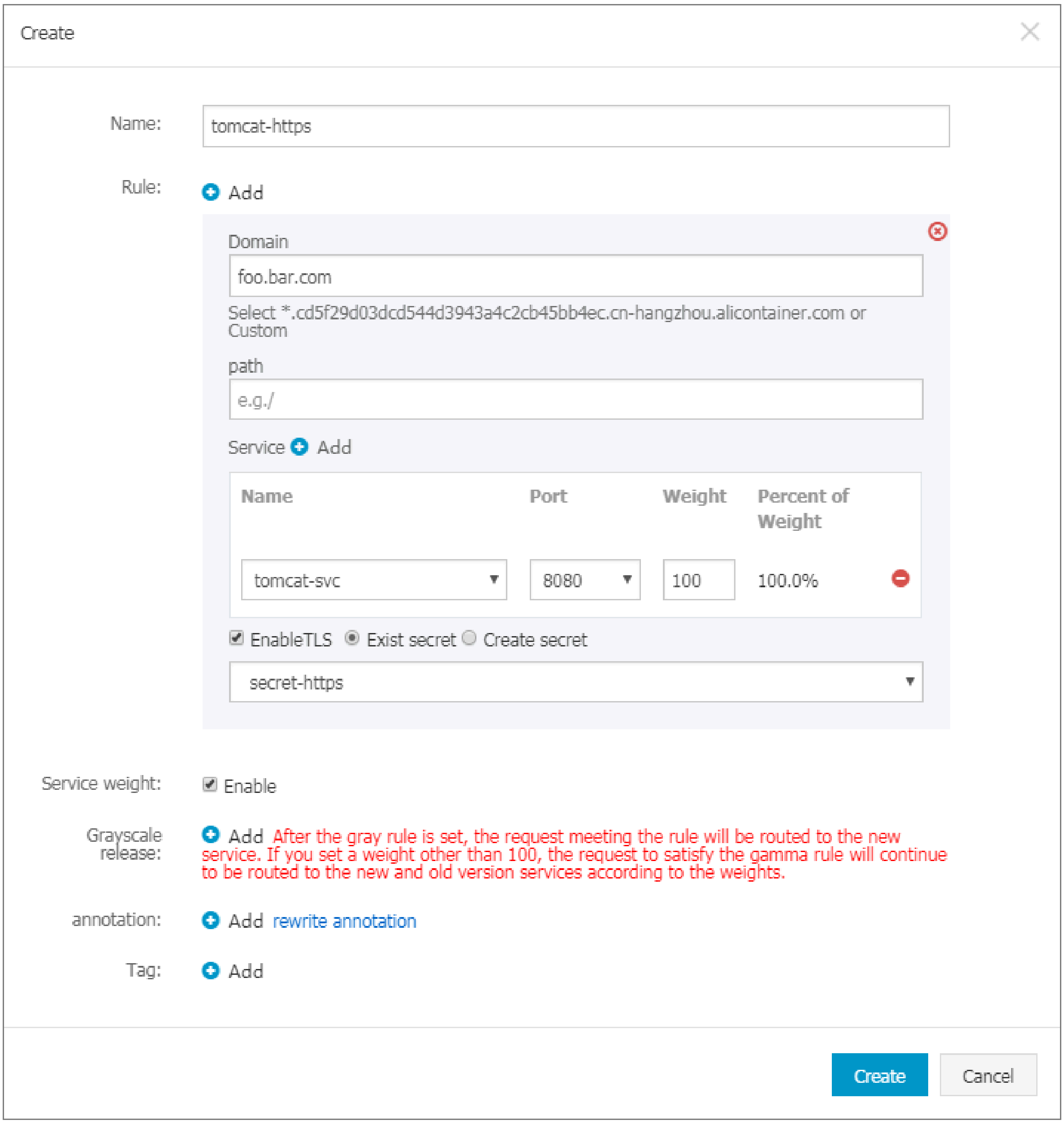
    Note:
    You must set the domain name properly. Otherwise, you will encounter exceptions when accessing the application through HTTPS.

    ```
    kubectl  create  secret  tls  secret - https  -- key  tls .
    key  -- cert  tls . crt
    ```
2. Log on to the Container Service console.
3. In the left-side navigation pane under Container Service-Kubernetes, choose Discovery and Load Balancing > Ingresses, select a cluster and namespace, and click Create in the upper-right corner.

4. In the displayed dialog box, configure the Ingress to make it accessible through HTTPS, and then click OK.

   For more information about Ingress configuration, see Create an Ingress in the Container Service console. The configuration in this example is as follows:

   - Name: Enter an Ingress name.
   - Domain: Enter the domain name set in the preceding steps. It must be the same as that configured in the SSL certificate.
   - Service: Select the service corresponding to the tomcat application.The service port is 8080.
   - Enable TLS: After enabling TLS, select the existing secret.



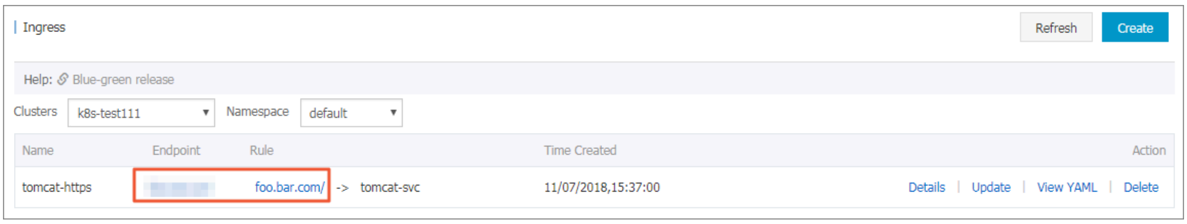
You can also use a YAML file to create an Ingress. In this example, the YAML sample file is as follows:


```
apiVersion :  extensions / v1beta1
kind :  Ingress
metadata :
   name :  tomcat - https
spec :
  tls :
 -  hosts :
   -  foo . bar . com
    secretName :  secret - https
  rules :
 -  host :  foo . bar . com
    http :
      paths :
    -  path : /
      backend :
        serviceNam  e :  tomcat - svc
        servicePor  t :  8080
```


5. Return to the Ingress list to view the created Ingress, the endpoint, and the domain name. In this example, the domain name is `foo . bar . com`. You can also enter the Ingress detail page to view the Ingress.

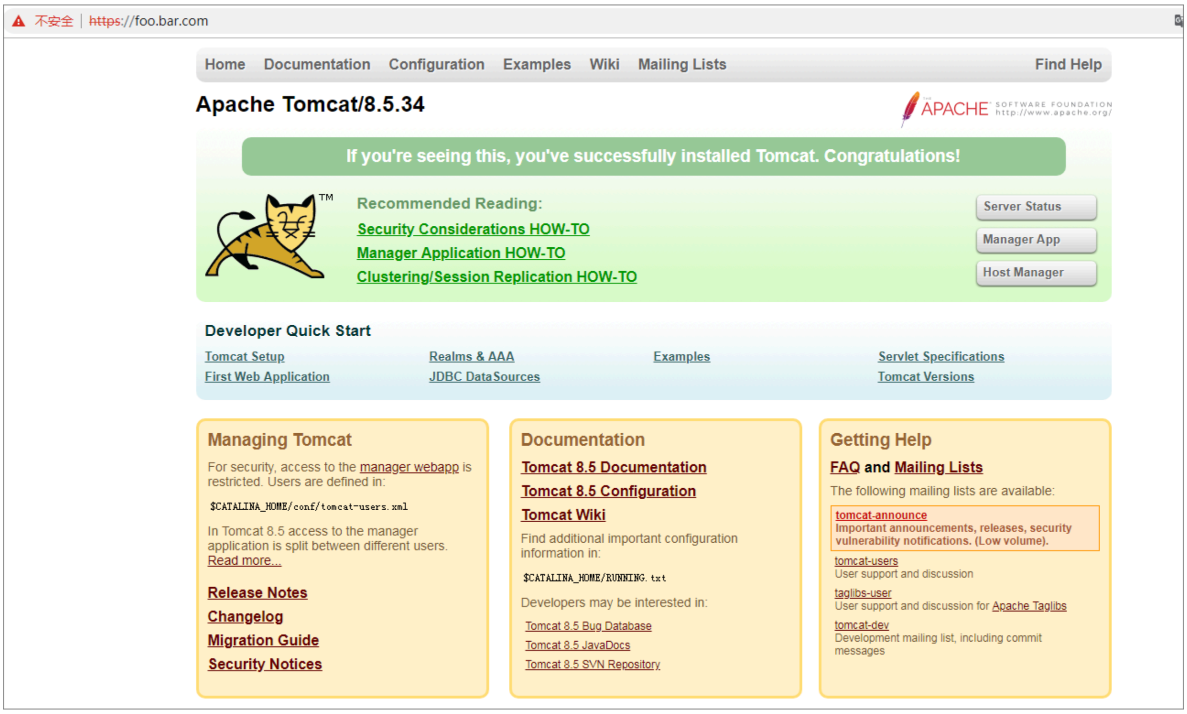
Note:
In this example, `foo . bar . com` is used as a testing domain name, and you need to create a record in the hosts file.

```
47 . 110 . 119 . 203     foo . bar . com                  # where
,  the   IP   address   is   the   Ingress   endpoint .
```



6. In the browser, access `https :// foo . bar . com` .

> **Note:**
> You need to access the domain name by using HTTPS because you have created a TLS access certificate. This example uses `foo . bar . com` as a sample domain name to be parsed locally. In your specific configuration scenarios, you need to use the registered domain names.



10.4 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 through the Container Service console.

Prerequisites

- A Kubernetes cluster is created with ACK and the cluster certificates are about to expire. For more information, see Create a Kubernetes cluster.
- The required tasks have been completed. For more information, see Cluster certificate update FAQ.

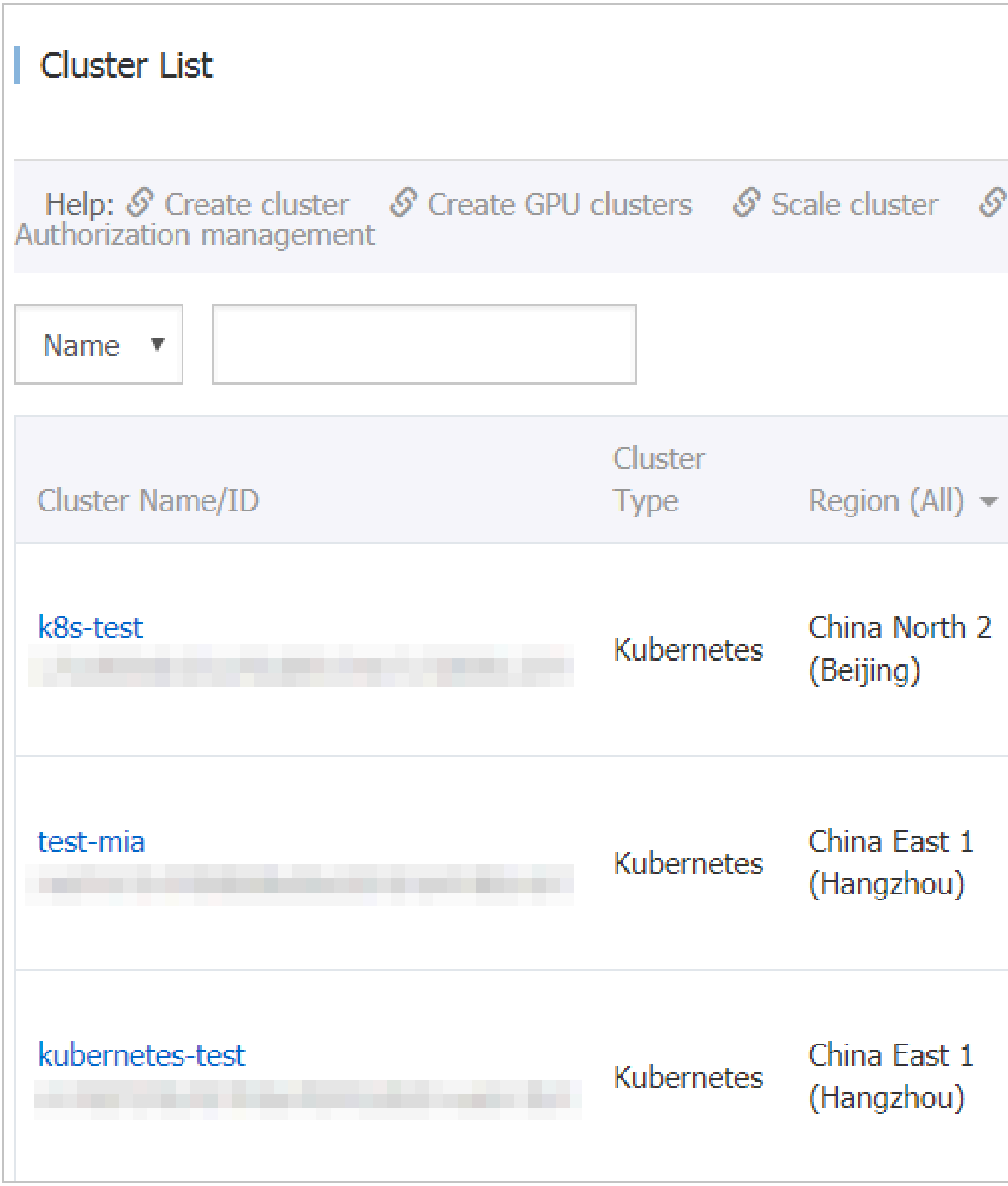
Procedure

1. Log on to the Container Service console.

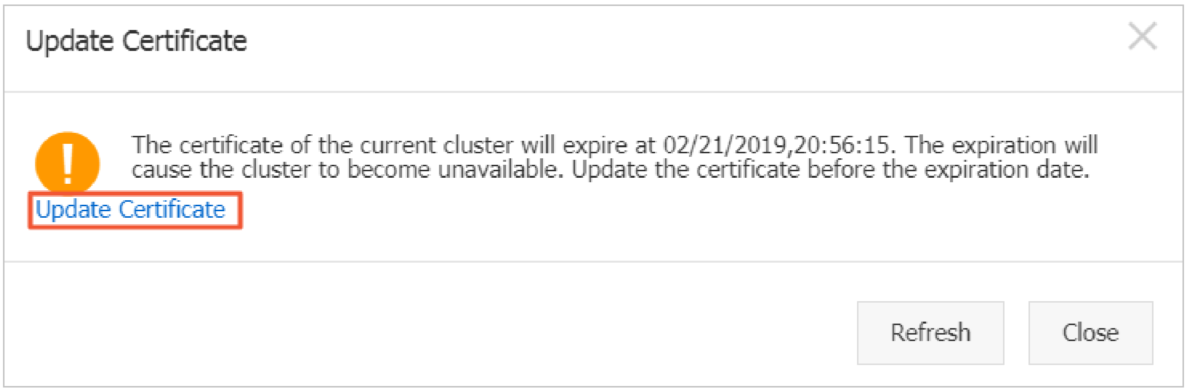
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.
3. Click Update Certificate on the right of the target cluster.

Note:

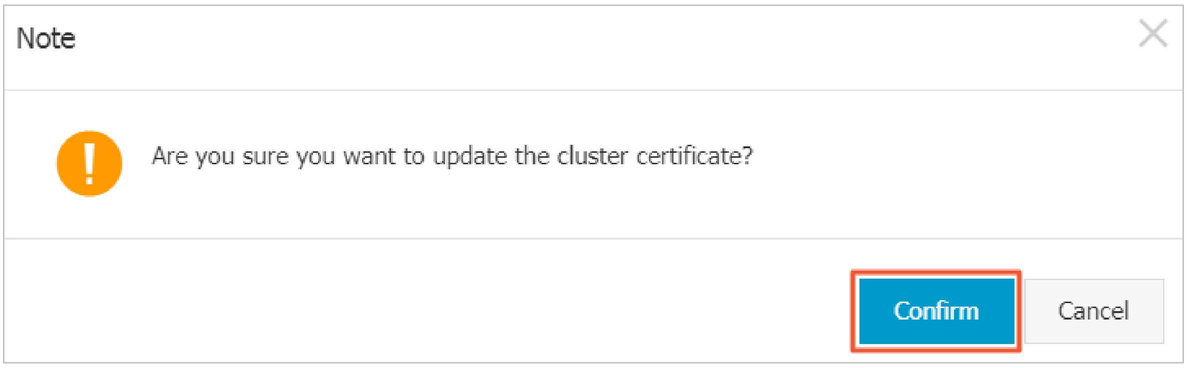
If cluster certificates are about to expire in about two months, the system displays the Update Certificate prompt for the cluster.



4. Click Update Certificate.

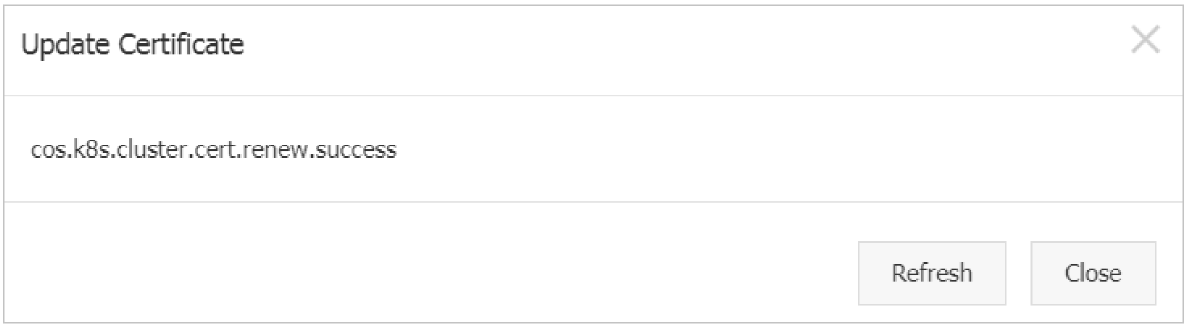


5. Click Confirm.



Result

- The Update Certificate page displays Success.



- On the Cluster List page, the Update Certificate prompt of the target cluster has been removed.

10.5 Security bulletins

10.5.1 Vulnerability fix: `CVE - 2019 - 5736` in runc

This topic describes the impacts of the security vulnerability `CVE - 2019 - 5736` in runc and how to remove it. This vulnerability has been fixed for the Kubernetes cluster versions 1.11 and 1.12.

Background

The security vulnerability may occur with Docker, containerd, or any other containers that use runc. This vulnerability gives attackers the ability to use a specific container image or run the exec command to obtain the file handle used by the running host runc. Attackers can overwrite the host runc binary file, then obtain root permission to access the host, and execute commands as with root permission.

For more information, see CVE-2019-5736.

Affected clusters

- Alibaba Cloud Container Service clusters affected by the vulnerability:
  - All Docker Swarm clusters from versions earlier than Docker v18.09.02.
  - All Kubernetes clusters except for Serverless Kubernetes clusters.
- Self-built Docker and Kubernetes clusters affected by the vulnerability:
  - All clusters that use Docker versions earlier than v18.09.2.
  - All clusters that use runc v1.0-rc6 or earlier.

Resolution

To fix the security CVE-2019-5736 vulnarability for Kubernetes clusters earlier than V1.11 and V1.12, use one of the following two methods:

- Upgrade the Docker version of all existing clusters to v18.09.2 or later. Using this method will interrupt your cluster containers and services.
- Only upgrade runc. This method is applicable to clusters running Docker v17.06. We recommend that you upgrade the runc binary file of each cluster node individually to avoid a service interruption caused by upgrading the Docker engine. To upgrade a runc binary file, complete the following steps:

  1. Run the following command to locate docker-runc:

     Note:

Usually, docker-runc is located in `/ usr / bin / docker - runc .`

```
which   docker - runc
```

2. Run the following command to back up the original runc:

```
mv  / usr / bin / docker - runc  / usr / bin / docker - runc .
orig .$( date  - Iseconds )
```

3. Run the following command to download the fixed runc:

```
curl  - o  / usr / bin / docker - runc  - sSL   https :// acs -
public - mirror . oss - cn - hangzhou . aliyuncs . com / runc /
docker - runc - 17 . 06 - amd64
```

4. Run the following command to set permission availability for docker-runc:

```
chmod  + x  / usr / bin / docker - runc
```

5. Run the following command to test whether runc works normally:

```
 docker - runc  - v
#  runc   version   1 . 0 . 0 - rc3
#  commit :  fc48a25bde  6fb041aae0  977111ad81  41ff396438
#  spec :  1 . 0 . 0 - rc5
 docker   run  - it  -- rm   ubuntu   echo   OK
```

6. To upgrade the runc binary file of a Kubernetes cluster GPU node, you must also install nvidia-runtime by completing the following steps:

   a. Run the following command to locate nvidia-container-runtime:

      Note:

      Usually, nvidia-container-runtime is located in `/ usr / bin / nvidia - container - runtime .`

      ```
      which   nvidia - container - runtime
      ```

   b. Run the following command to back up the original nvidia-container-runtime:

      ```
      mv  / usr / bin / nvidia - container - runtime  / usr / bin /
      nvidia - container - runtime . orig .$( date  - Iseconds )
      ```

   c. Run the following command to download the fixed nvidia-container-runtime:

      ```
      curl  - o  / usr / bin / nvidia - container - runtime  - sSL
        https :// acs - public - mirror . oss - cn - hangzhou .
      ```

```
aliyuncs . com / runc / nvidia - container - runtime - 17 . 06
- amd64
```

d. Run the following command to set permission availability for nvidia-container-runtime:

```
chmod  + x  / usr / bin / nvidia - container - runtime
```

e. Run the following command to test whether nvidia-container-runtime works normally:

```
 nvidia - container - runtime  - v
#   runc   version   1 . 0 . 0 - rc3
#   commit :  fc48a25bde  6fb041aae0  977111ad81  41ff396438 -
 dirty
#   spec :  1 . 0 . 0 - rc5

 docker   run  - it  -- rm  - e   NVIDIA_VIS  IBLE_DEVIC  ES =
 all   ubuntu   nvidia - smi  - L
#   GPU   0 :  Tesla   P100 - PCIE - 16GB  ( UUID :  GPU -
 122e199c - 9aa6 - 5063 - 0fd2 - da009017e6  dc )
```

Note:
This test is performed on a node of the GPU P100 model. Test outputs vary by GPU models.

10.5.2 Vulnerability fix: *CVE - 2018 - 18264* for Kubernetes dashboard

Alibaba Cloud Container Service for Kubernetes has fixed dashboard vulnerability *CVE - 2018 - 18264* . This topic describes the dashboard versions affected by the vulnerability and how to fix the vulnerability. The Kubernetes dashboards that are built in Alibaba Cloud Container Service for Kubernetes are not affected by this vulnerability because they work in the hosted form and their security settings were upgraded before the vulnerability occurred.

Background information

A security vulnerability, that is, *CVE - 2018 - 18264* , was discovered in Kubernetes dashboards of V1.10 and earlier versions. This vulnerability allowed attackers to bypass identity authentication and read secrets within the cluster by using the dashboard logon account.

The Kubernetes dashboards that are built in Alibaba Cloud Container Service for Kubernetes are not affected by this vulnerability because they work in the hosted form and their security settings were upgraded before the vulnerability occurred.

For more information about security vulnerability `CVE - 2018 - 18264` , see:

- https://github.com/kubernetes/dashboard/pull/3289
- https://github.com/kubernetes/dashboard/pull/3400
- https://github.com/kubernetes/dashboard/releases/tag/v1.10.1

Conditions required to determine that a Kubernetes dashboard is vulnerable

Your dashboard is vulnerable if you have independently deployed Kubernetes dashboard V1.10 or earlier versions (V1.7.0 to V1.10.0) that supports the logon function in your Kubernetes cluster, and you have used custom certificates.

Resolution

- If you do not need a dashboard that is deployed independently, run the following command to remove the Kubernetes dashboard from your cluster:

```
kubectl  -- namespace  kube - system  delete  deployment
kubernetes - dashboard
```

- If you need an independently deployed dashboard, upgrade your dashboard to V1.10.1. For more information, see https://github.com/kubernetes/dashboard/releases/tag/v1.10.1.
- If you use the dashboard hosted by Alibaba Cloud Container Service for Kubernetes, you can continue to use your dashboard in the Container Service console because the dashboard was upgraded before the vulnerability occurred.

10.5.3 Vulnerability fix: `CVE - 2018 - 1002105`

Alibaba Cloud has fixed system vulnerability `CVE - 2018 - 1002105` . This topic describes the impacts of this vulnerability and how to remove it.

This vulnerability does not affect Serverless Kubernetes clusters. Serverless Kubernetes was upgraded before the vulnerability occurred.

Background information

Engineers of the Kubernetes community have found security vulnerability `CVE - 2018 - 1002105` . Kubernetes users can gain access to the backend service by forging the request and escalating the permission on the established API Server connection. Alibaba Cloud has fixed this vulnerability. To remove the vulnerability, you need to log on to the Container Service console and upgrade Kubernetes to the latest version.

For more information about the vulnerability `CVE - 2018 - 1002105` , see https://github.com/kubernetes/kubernetes/issues/71411.

Affected Kubernetes versions:

- Kubernetes v1.0.x-1.9.x
- Kubernetes v1.10.0-1.10.10 (fixed in v1.10.11)
- Kubernetes v1.11.0-1.11.4 (fixed in v1.11.5)
- Kubernetes v1.12.0-1.12.2 (fixed in v1.12.3)

Affected configurations:

- Kubernetes cluster, which runs on Container Service and uses an extension API server. Furthermore, the extension API server network is directly accessible to the cluster component, kube-apiserver.
- Kubernetes cluster, which runs on Container Service and has opened permission s to interfaces such as pod exec, attach, and portforward. Then, users can use the vulnerability to obtain permissions to access all kubelet APIs of the cluster.

Cluster configuration of Alibaba Cloud Container Service for Kubernetes

- The API server of a Kubernetes cluster that runs on Container Service has RBAC enabled by default. That is, the API server denies anonymous user access through primary account authorization. Furthermore, the starting parameter of Kubelet is `anonymous - auth = false` , providing security access control against external attacks.
- If your Kubernetes cluster has multiple RAM users, the RAM users may gain unauthorized access to the backend service through interfaces such as pod exec , attach, and portforward. If your cluster has no RAM users, you do not need to worry about the vulnerability.
- RAM users do not have access to aggregate API resources by default without custom authorization from the primary account.

Solution

Log on to the Container Service console to upgrade your cluster. For more information, see #unique_147.

- If your cluster is V1.11.2, upgrade it to V1.11.5.
- If your cluster is V1.10.4, upgrade it to V1.10.11 or V1.11.5.

- If your cluster is V1.9 or earlier, upgrade it to V1.10.11 or V1.11.5. When you upgrade the cluster from V1.9 to V1.10 or V1.11, upgrade the flexvolume plugin through the console if your cluster uses cloud disk volumes.

> Note:
> In the Container Service console, select the target cluster and choose More > Addon Upgrade. In the Addon Upgrade dialog box, select flexvolume and click Upgrade.

10.5.4 Vulnerability fix: `CVE - 2019 - 11246` related to kubectl cp

This topic describes how to fix the `CVE - 2019 - 11246` vulnerability related to the `kubectl cp` command.

Background information

The `kubectl cp` command is used to copy files between containers and hosts. When you copy a file from a container to your host, Kubernetes first runs the `tar` command to create a corresponding archive file and sends the archive file to your host. Then, kubectl decompresses the archive file on your host.

The `CVE - 2019 - 11246` vulnerability provides attackers the opportunity to write malicious files saved in a TAR package into any paths on your host by running the `kubectl cp` command through path traversal.

If the TAR package contains malicious files, attackers who have the permission to run the `kubectl cp` command can perform path traversal.

The effects of this vulnerability are similar to those of the `CVE - 2019 - 1002101` vulnerability. For information about the `CVE - 2019 - 1002101` vulnerability, see CVE-2019-1002101: kubectl fix potential directory traversal.

For information about the vulnerability PR, see CVE-2019-11246: Clean links handling in cp's tar code.

For more information about security issues caused by this vulnerability, see kubernetes-security-announce.

Affected Kubernetes versions

- kubectl v1.11.x and earlier versions
- kubectl v1.12.1-v1.12.8 (fixed in v1.12.9)

- kubectl v1.13.1-v1.13.5 (fixed in v1.13.6)
- kubectl v1.14.1 (fixed in v1.14.2)

> **Note:**
> You can view the versions of kubectl by running the `kubectl version --client` command.

Solution

Upgrade kubectl and confirm the kubectl version. For more information, see Install and set up kubectl.

- If your kubectl is v1.12.x, upgrade it to v1.12.9.
- If your kubectl is v1.13.x, upgrade it to v1.13.6.
- If your kubectl is v1.14.x, upgrade it to v1.14.2.
- If your kubectl is v1.11 or an earlier version, upgrade it to v1.12.9, v1.13.6, or v1.14.2.

11 Release management

11.1 Manage a Helm-based release

Alibaba Cloud Container Service for Kubernetes is integrated with the package management tool Helm to help you quickly deploy applications on the cloud. However, Helm charts can be released multiple times and the release version must be managed. Container Service for Kubernetes provides a release function, which allows you to manage the applications released by using Helm in the Container Service console.

Prerequisites

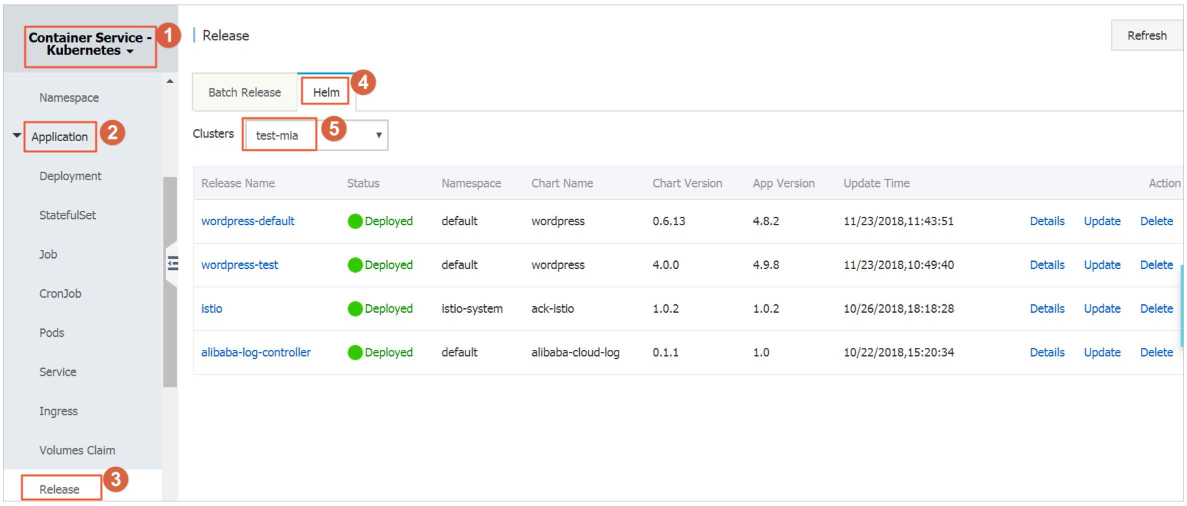
- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have installed a Helm application by using the App Catalog function or Service Catalog function. For more information, see Simplify Kubernetes application deployment by using Helm. In this topic, the wordpress-default application is used as an example.

View release details

1. Log on to the Container Service console.

2. In the left-side navigation pane, select Container Service - Kubernetes. Then, select Application > Release and click the Helm tab. Select the target cluster from the Clusters drop-down list.

   In the displayed release list, you can view the applications and services released through Helm in the selected cluster.

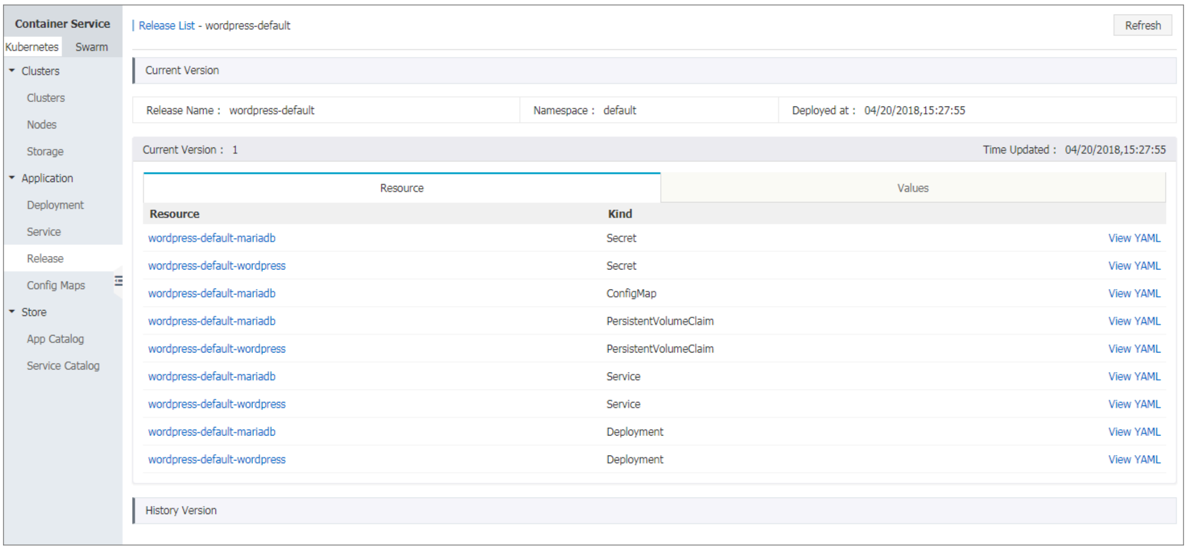


3. Find your target release (wordpress-default in this example) and click Details to view the release details.

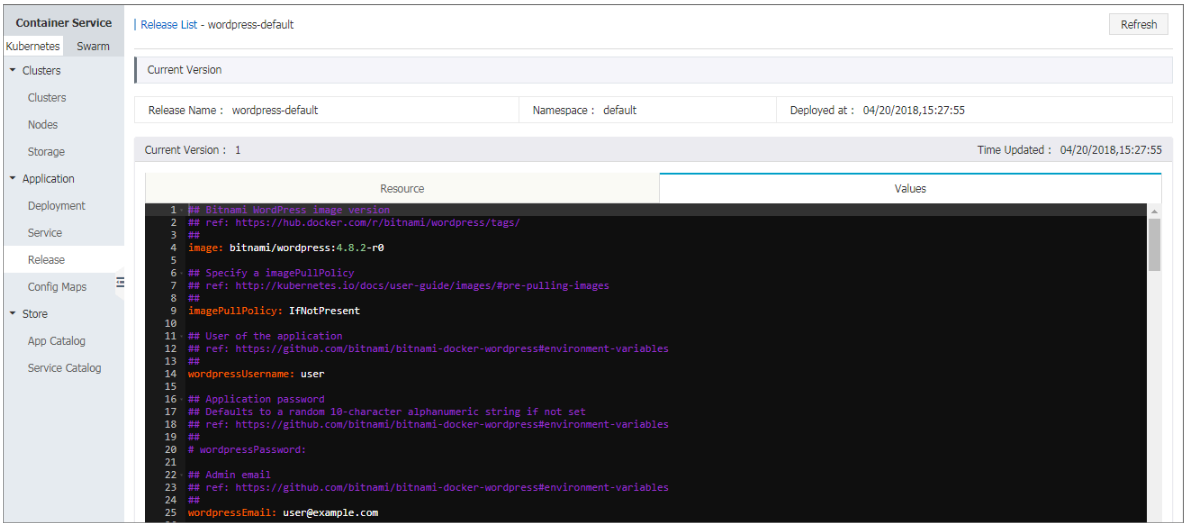
   You can view such release details as the current version and history version. In this example, the current version is 1 and no history version exists. On the Resource tab page, you can view the resource information of wordpress-default, such as the resource name and the resource type, and view the YAML information.

   Note:

You can view the running status of the resource in details by clicking the resource name and going to the Kubernetes dashboard page.



4. Click the Values tab to view the release parameters.

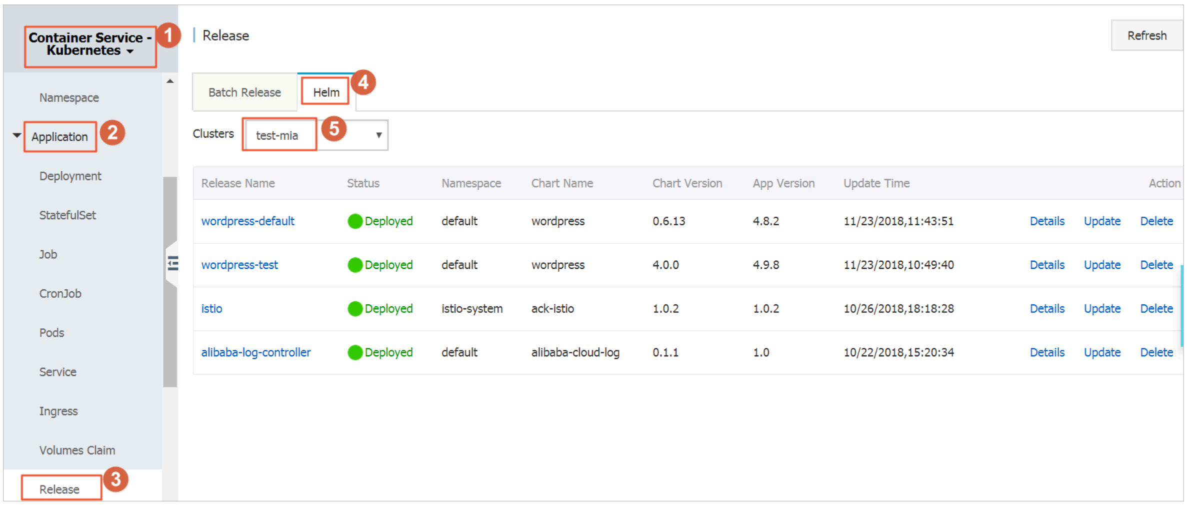


Update a release version

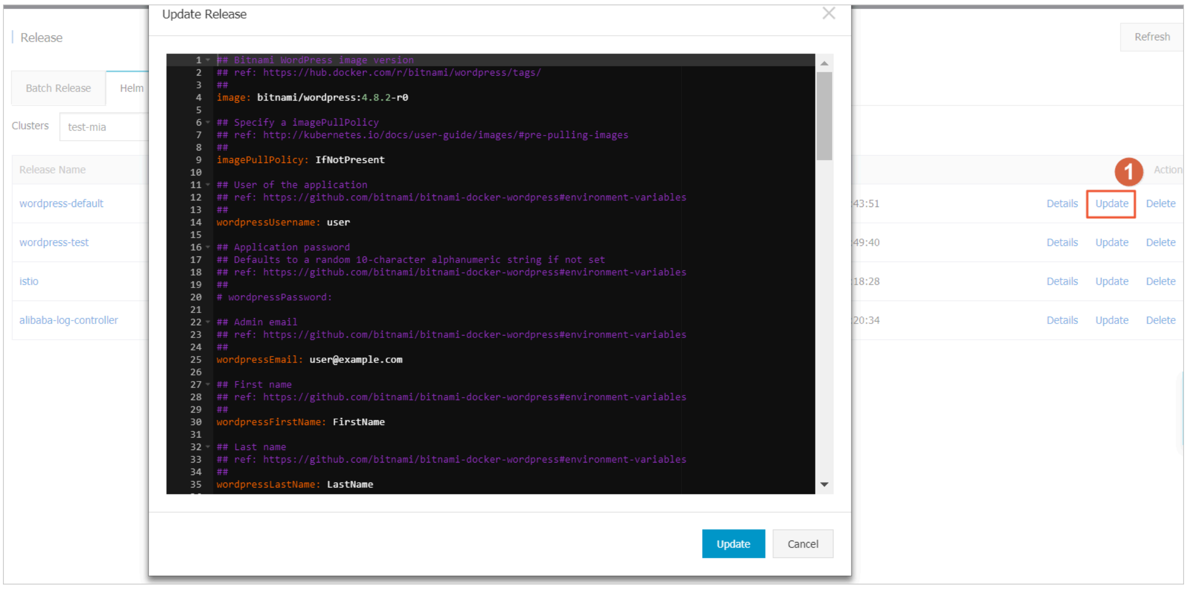
1. Log on to the Container Service console.

2. In the left-side navigation pane, select Container Service - Kubernetes. Then, select Application > Release and click the Helm tab. Select the target cluster from the Clusters drop-down list.

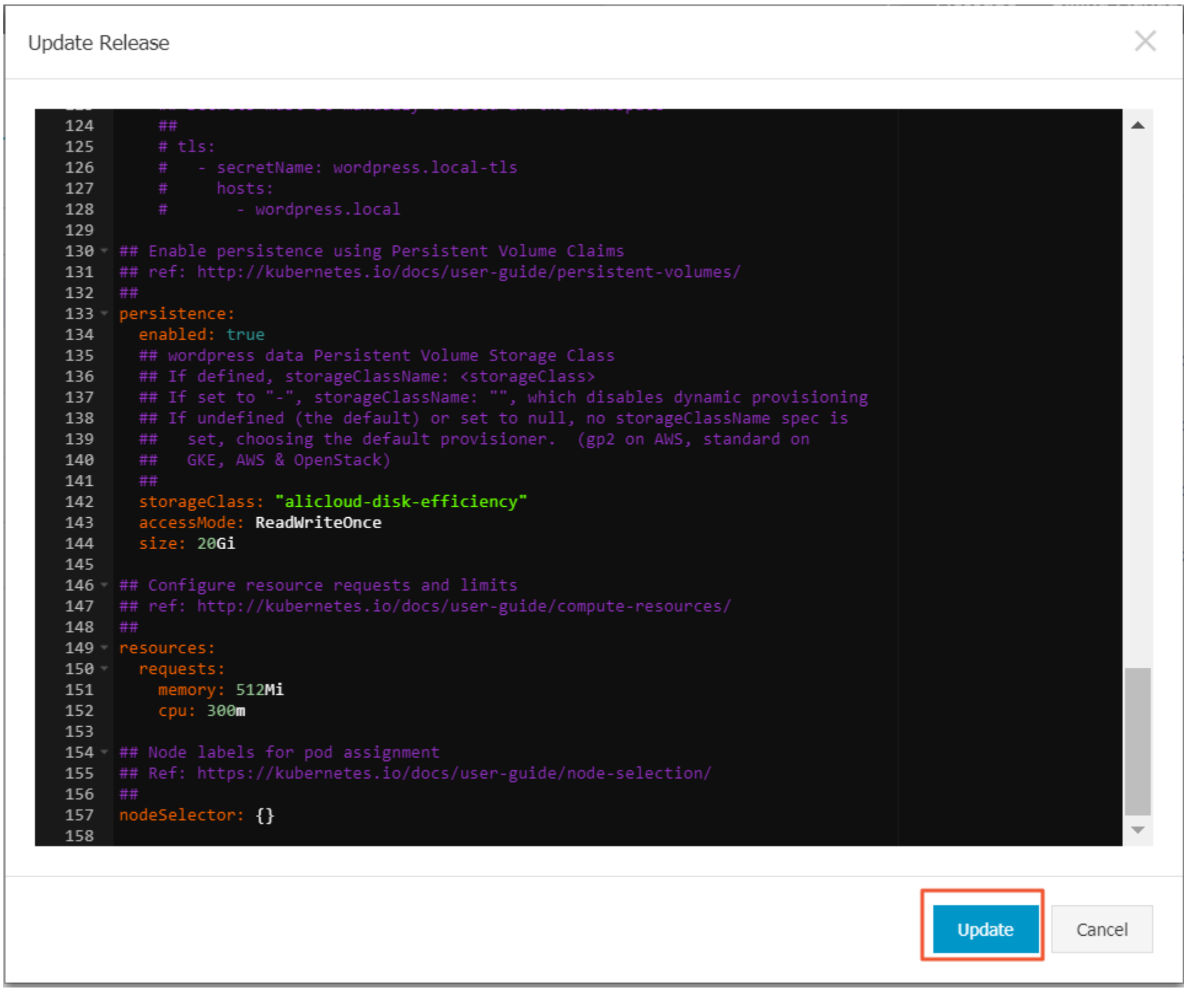
   In the displayed release list, you can view the applications and services released through Helm in the selected cluster.



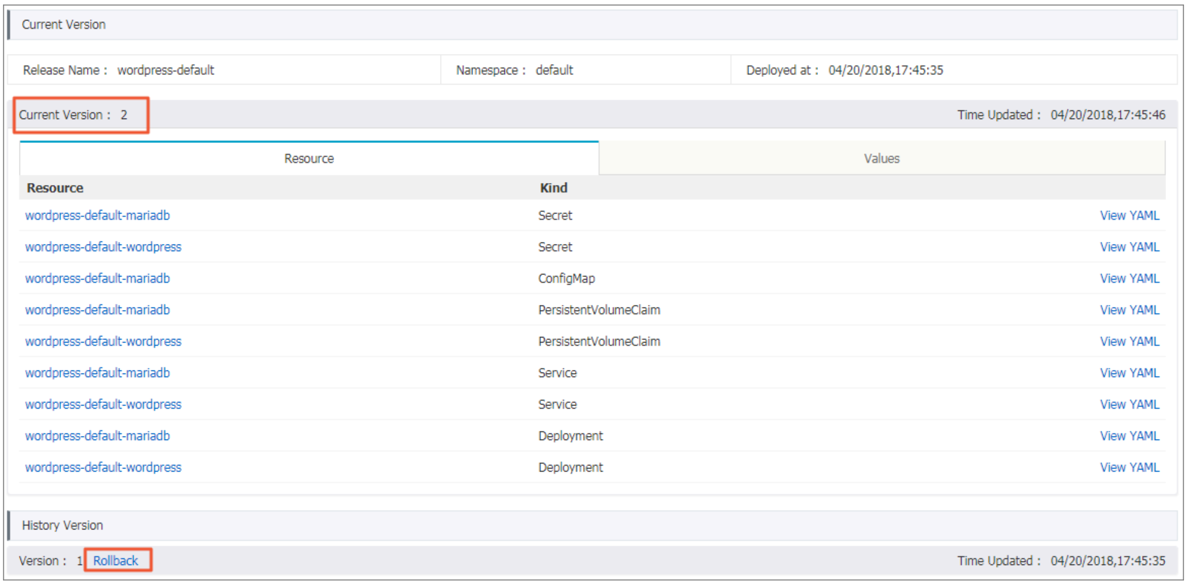
3. Find your target release (wordpress-default in this example). Click Update and the Update Release dialog box appears.



4. Modify the parameters and then click Update.



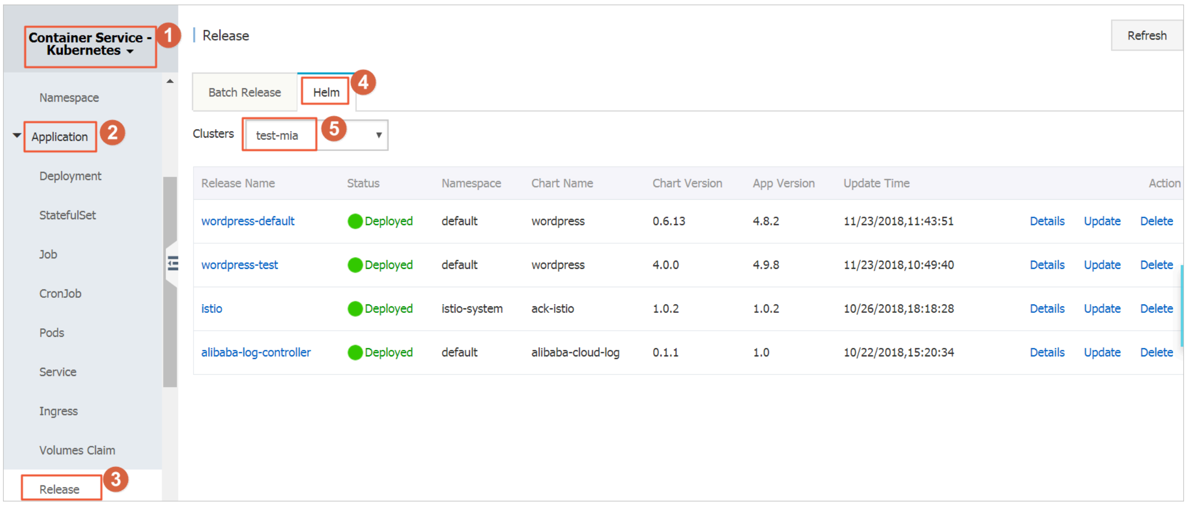
On the release list page, you can see that the current version changes to 2. To roll back to version 1, click Details and in the History Version area, click Rollback.



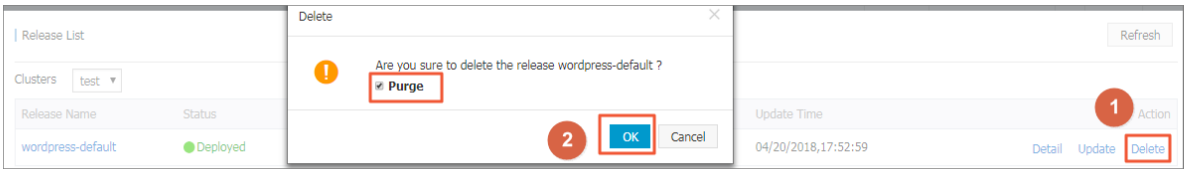
Delete a release

1. Log on to the Container Service console.
2. In the left-side navigation pane, select Container Service - Kubernetes. Then, select Application > Release and click the Helm tab. Select the target cluster from the Clusters drop-down list.

   In the displayed release list, you can view the applications and services released through Helm in the selected cluster.



3. Find your target release (wordpress-default in this example). Click Delete and the Delete dialog box appears.



4. Select the Purge check box if you want to clear the release records, and then click OK. After you delete a release, the related resources such as the services and deployments are deleted too.

11.2 Use batch release on Alibaba Cloud Container Service for Kubernetes

You can use Alibaba Cloud Container Service for Kubernetes to release application versions in batches, achieving fast version verification and rapid iteration of applications.

Context

Note:

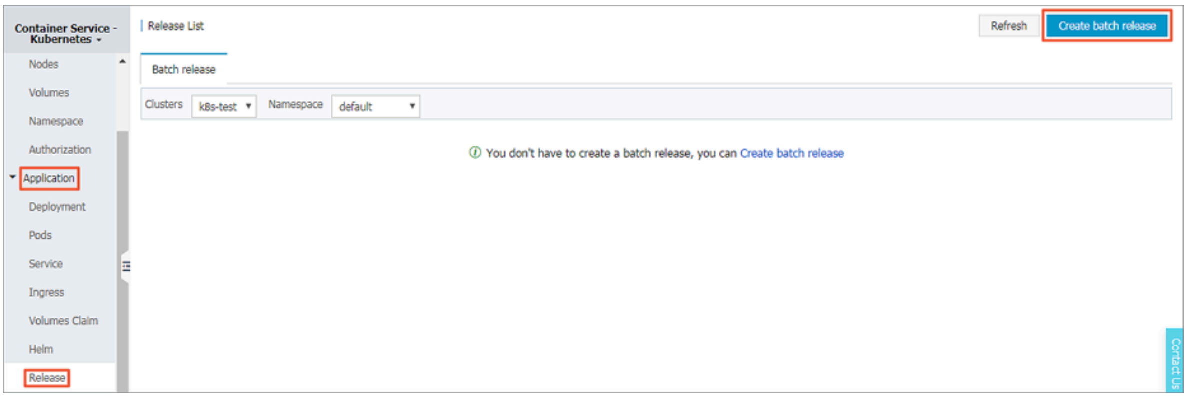
The latest Kubernetes cluster has installed alicloud-application-controller by default. For older versions of clusters, only versions of 1.9.3 and later are currently supported, and you can upgrade old versions of clusters through the prompt link on the console.

Procedure

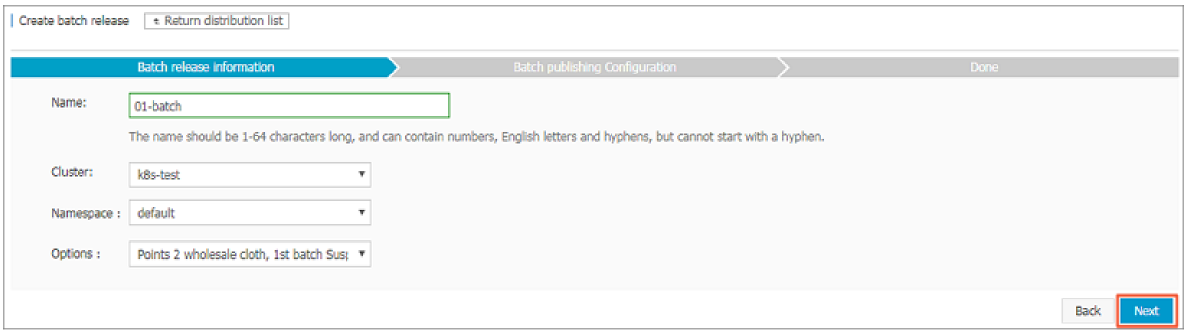
1. Log on to the Container Service console.
2. Under Kubernetes, click Application > Release in the left-side navigation pane. Click Create batch release in the upper-right corner.

Note:

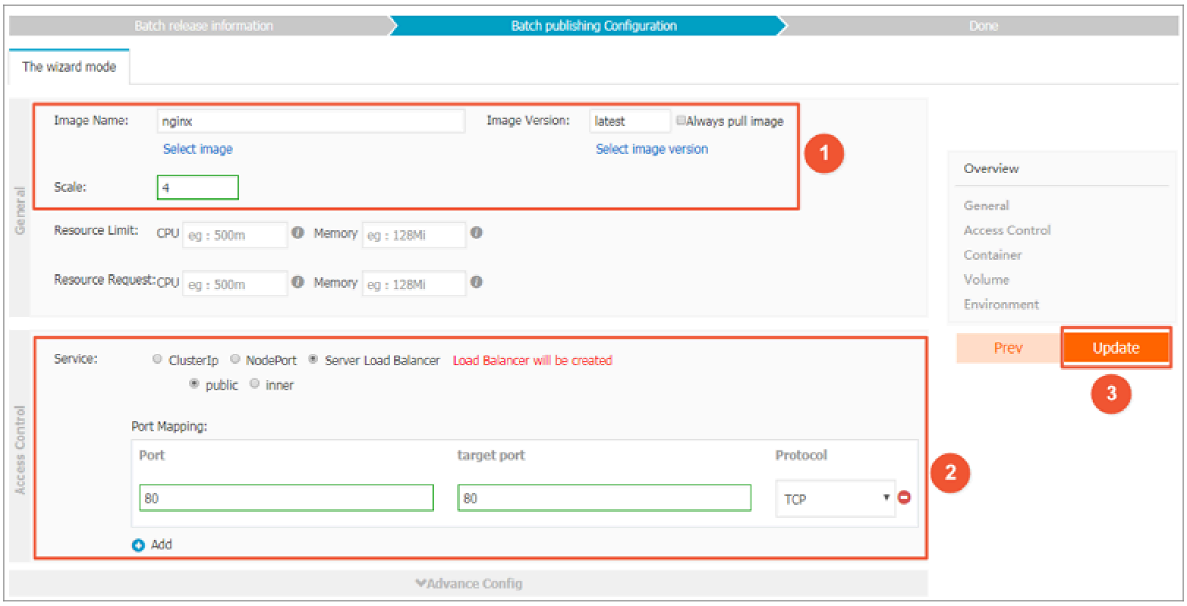
If the button is gray, you can upgrade the cluster by following the upgrade link.



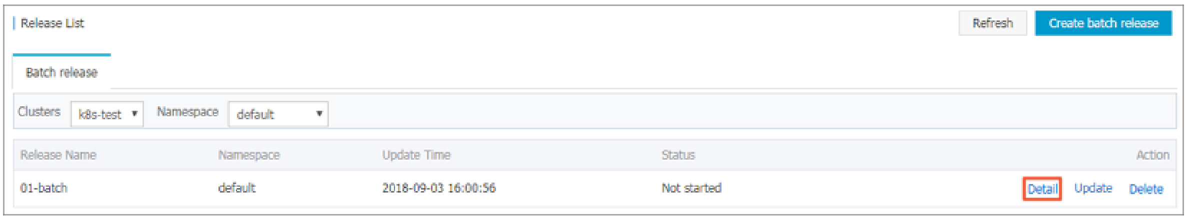
3. Configure batch release information, including the application name, cluster, namespace, and options. Click Next.



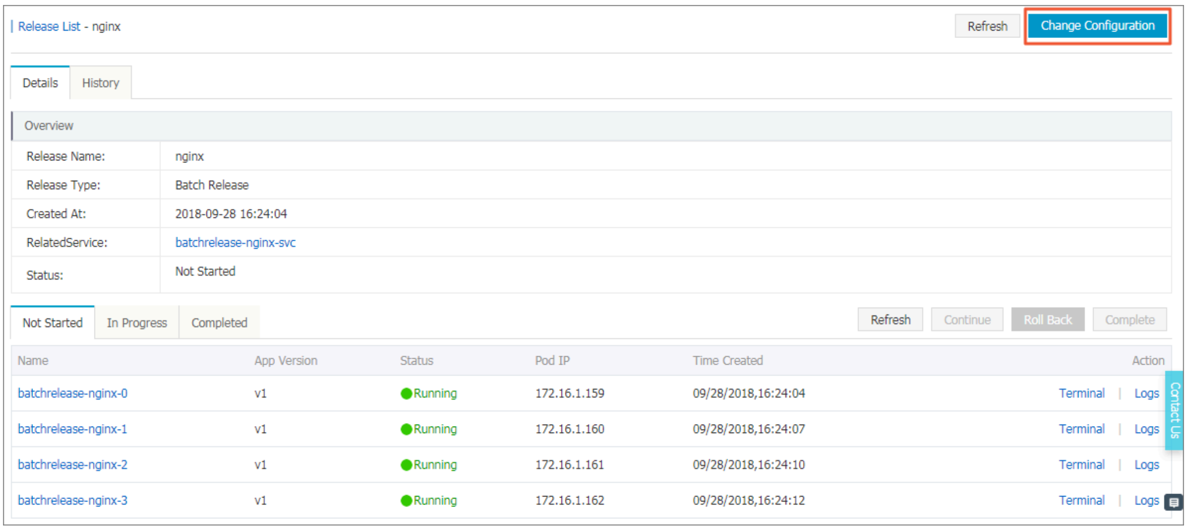
4. On the batch publishing configuration page, configure the backend pod and service, and then click Update to create an application.



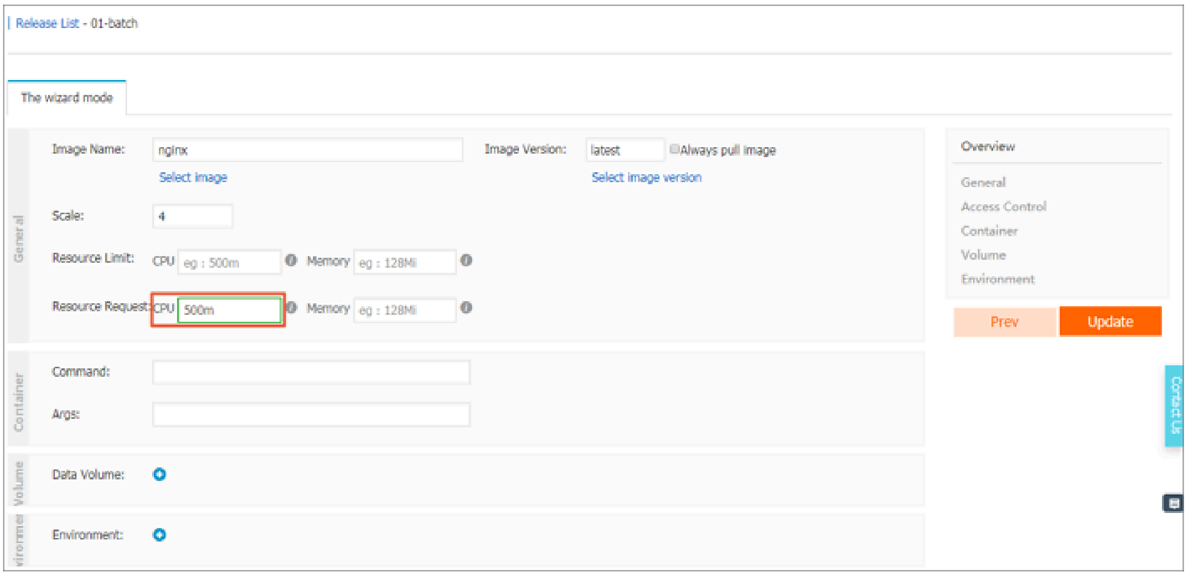
5. Return to the release list, an application is displayed in the Not started status. Click Detail on the right.



6. On the application detail page, you can view more information. Click Change Configuration in the upper-right corner of the page to make a batch release change.

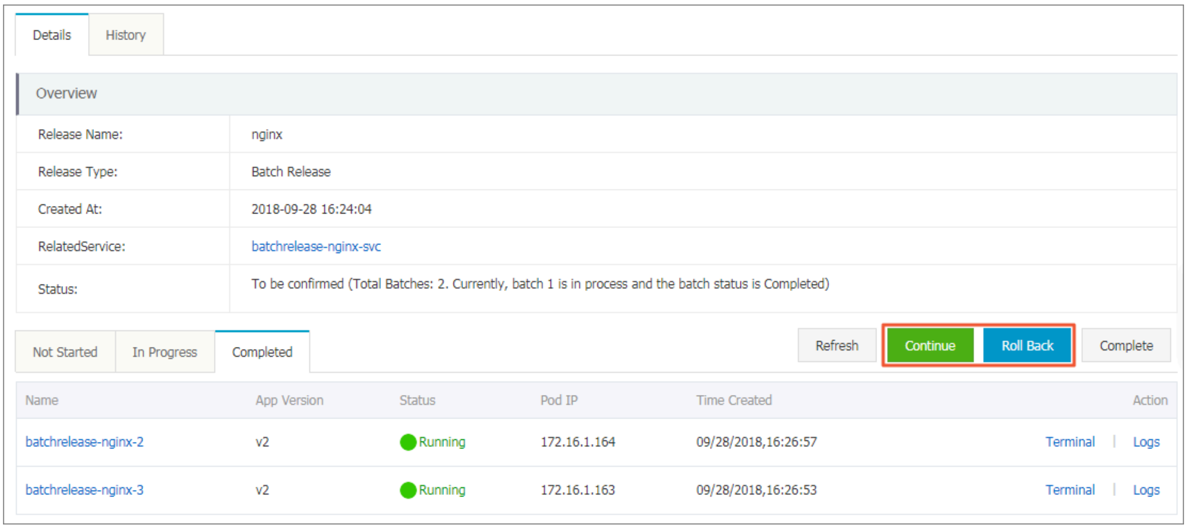


7. Configure changes for the new version of the application, and then click Update.

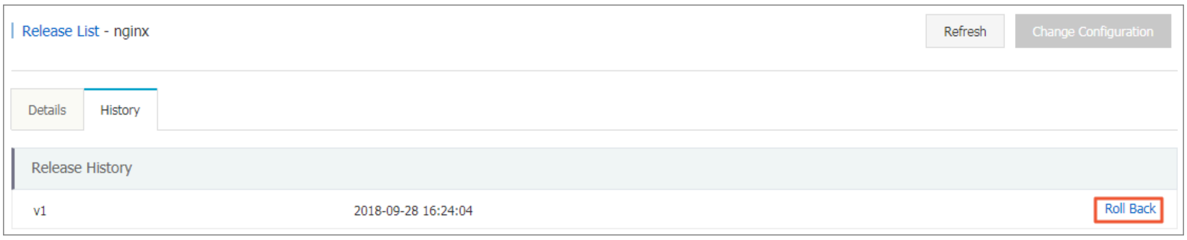


8. By default, you return to the release list page, where you can view the batch release status of the application. After completing the first deployment, click Detail.

9. You can see that the Not Started list is has two pods and the Completed list has two pods, which indicates that the first batch has been completed in batch release. Click Continue, you can release the second batch of pods. Click Roll Back to roll back to the previous version.



10.When completing the release, click Historyto roll back to history versions.



What's next

You can use batch release to quickly verify your application version without traffic consumption. Batch release is more resource-saving than blue-green release. Currently, batch release can be performed on only web pages. The yaml file editing is to be opened later to support more complex operations.

12 Namespace management

12.1 Create a namespace

This topic describes how to create a namespace.

Prerequisites

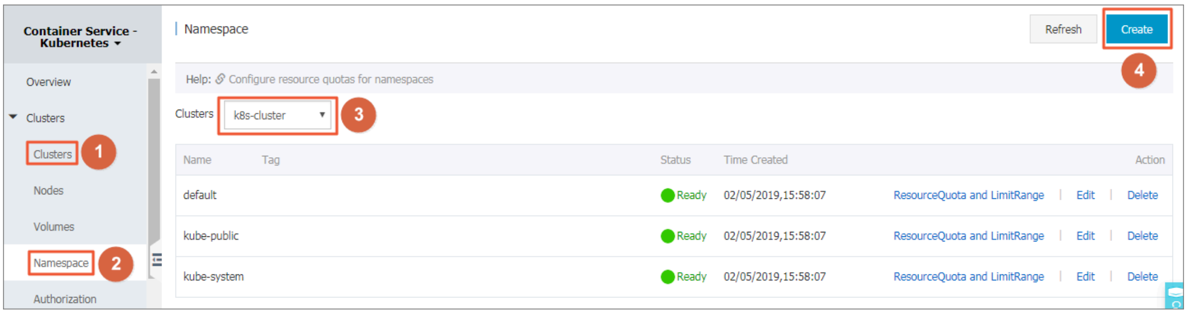
You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.

Context

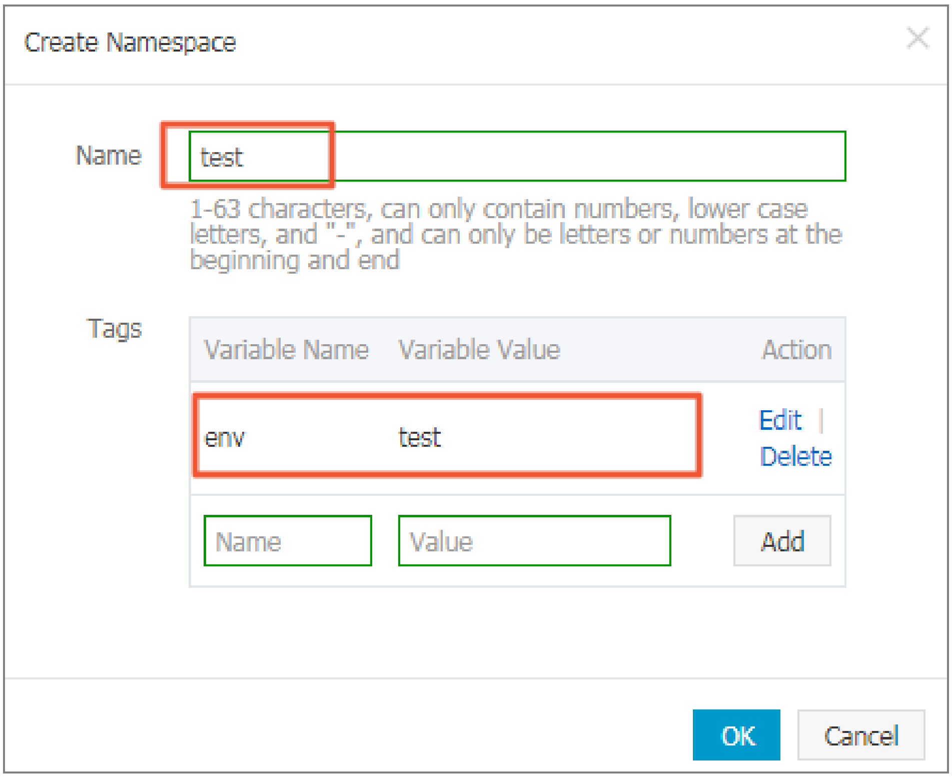
In a Kubernetes cluster, you can use namespaces to create multiple virtual spaces. When a large number of users share a cluster, multiple namespaces can be used to effectively divide different work spaces and assign cluster resources to different tasks. Furthermore, you can use resource quotas to assign resources to each namespace.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Namespace.
3. Select the target cluster, and then click Create in the upper-right corner.



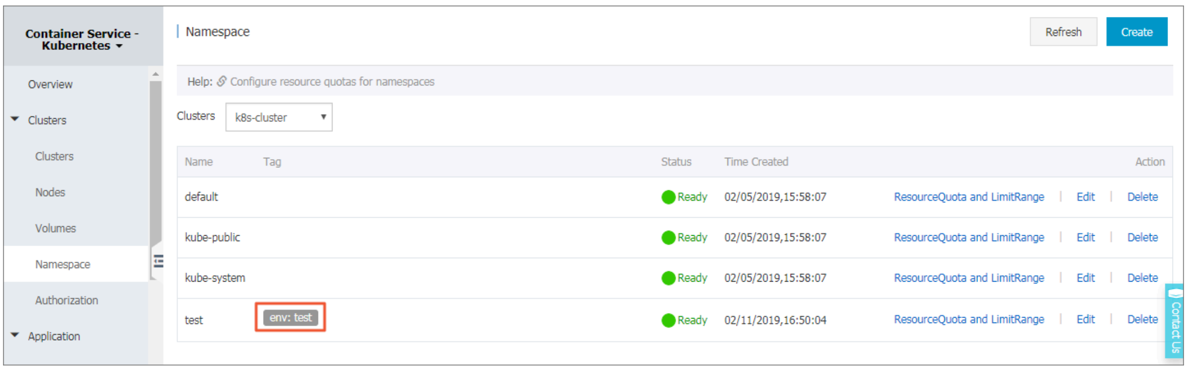
4. In the displayed dialog box, set a namespace.



- Name: Enter a name for the namespace name. The name must be 1 to 63 characters in length and can contain numbers, letters, and hyphens (-). It must start and end with a letter or number. In this example, test is used as the name.
- Tags: Add one or multiple tags to the namespace to identify the characteristics of the namespace. For example, you can set a tag to identify that this namespace is used for the test environment.

  You can enter a variable name and a variable value, and then click Add on the right to add a tag to the namespace.

5. Click OK.
6. The namespace named test is displayed in the namespace list.



12.2 Set resource quotas and limits for a namespace

This topic describes how to set resource quotas and limits for a namespace through the Container Services console.

Prerequisites

- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have created a namespace. In this topic, a namespace named test is used. For more information, see Create a namespace.
- You have connected to the Master node of the cluster. For more information, see Connect to a Kubernetes cluster by using kubectl.

Context

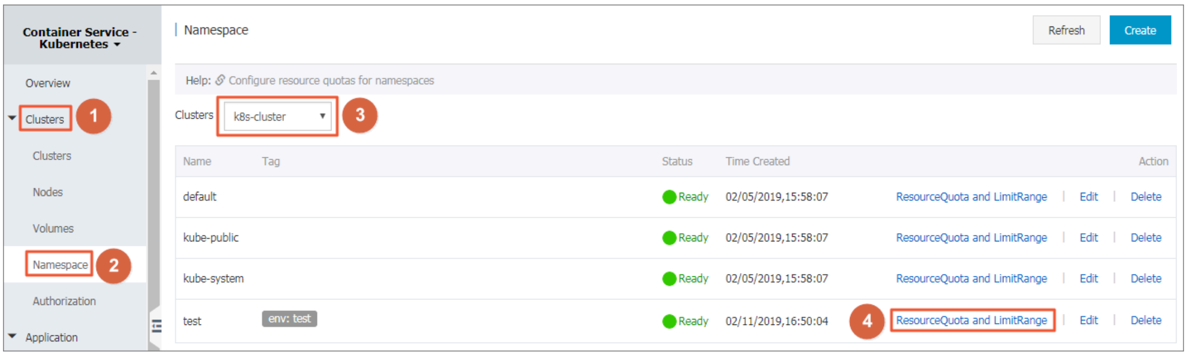
By default, a running pod uses the CPU and memory resources of nodes without limit . That is, any pod can use the computing resources of the cluster without restraint. Therefore, pods of a namespace may deplete the cluster resources.

Namespaces can be used as virtual clusters to serve multiple users. Therefore, setting resource quotas for a namespace is regarded as a best practice.

For a namespace, you can set the quotas of resources, such as CPU, memory, and number of pods. For more information, see Resource quotas.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Namespace. Select the target cluster and click ResourceQuota and LimitRange on the right of the test namespace.

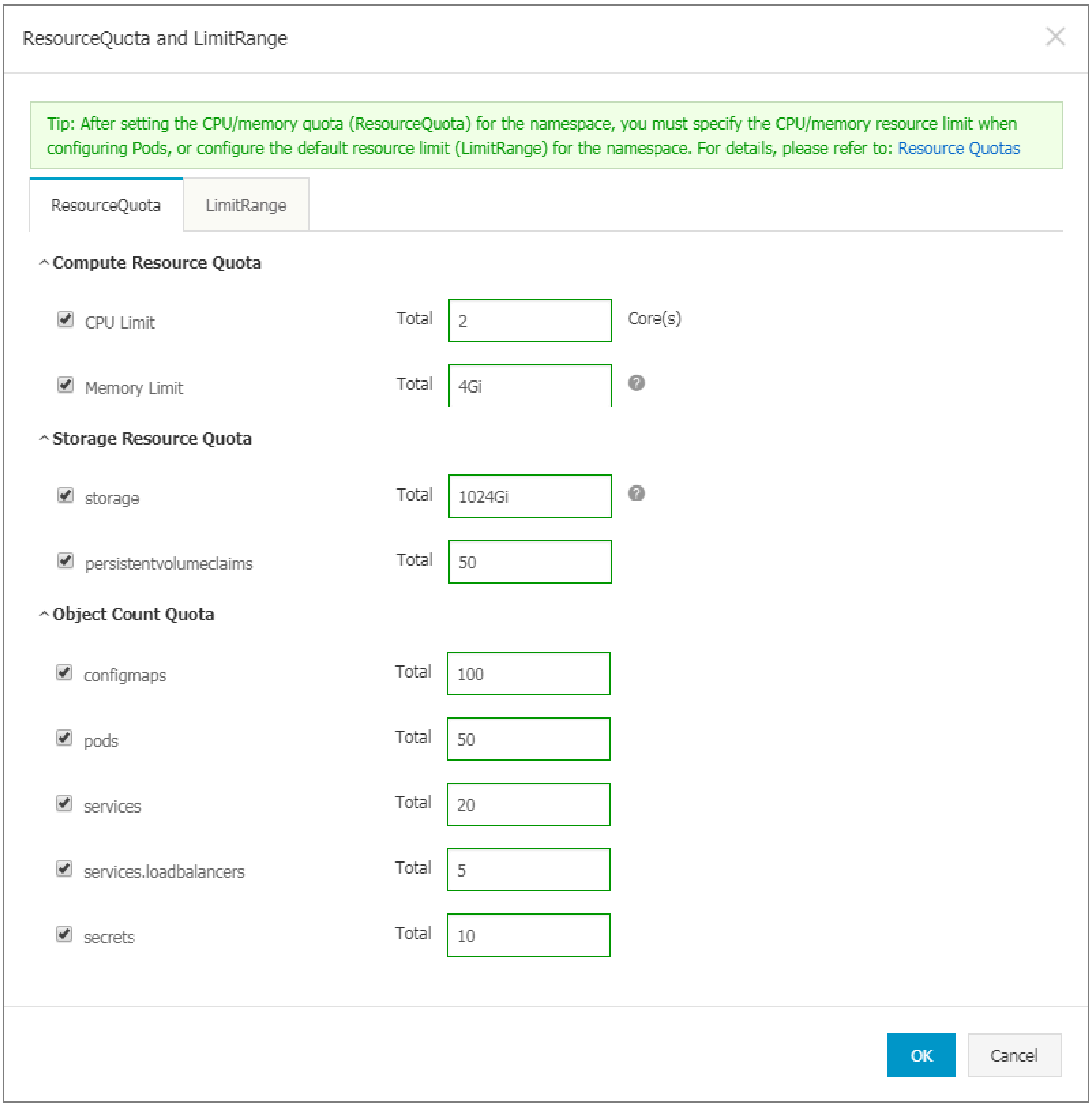


3. In the displayed dialog box, set resource quotas and default resource limits.

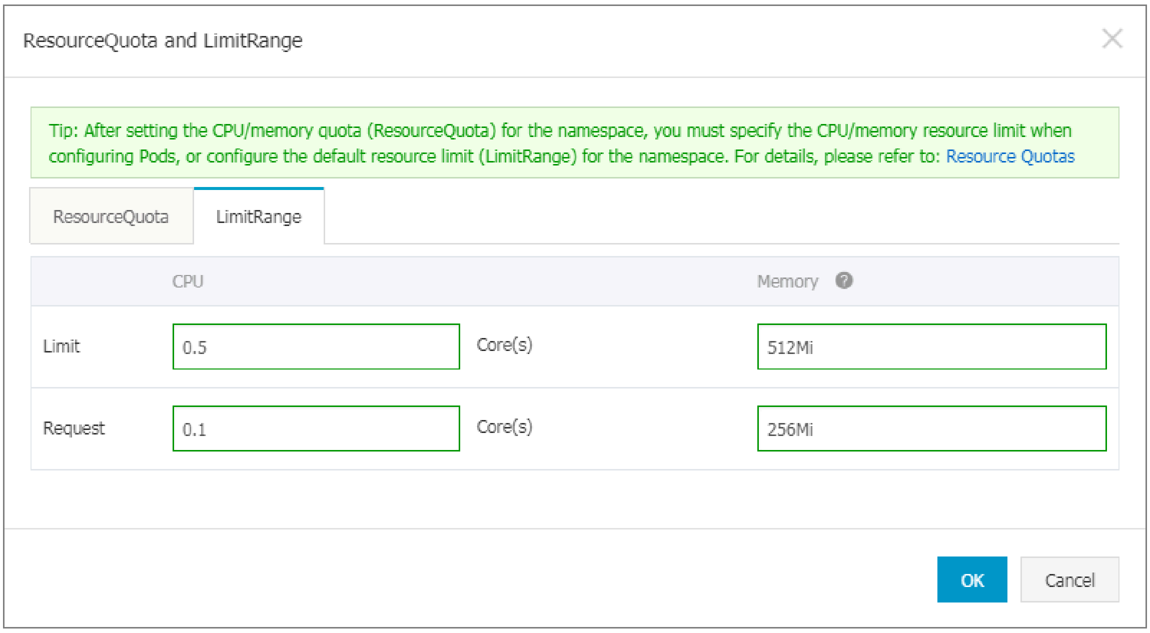
Note:

After setting CPU/memory quotas for a namespace, you must specify CPU/memory resource limits or set the default resource limits for the namespace when creating a pod. For more information, see Resource quotas.

a) Set resource quotas for the namespace.



b) To control the amount of resources consumed by containers, set resource limits and resource requests for containers in this namespace. For more information, see https://kubernetes.io//memory-default-namespace/.



4. Connect to the Master node and then run the following commands to view the resources of the test namespace:

```
# kubectl   get   limitrange , ResourceQu  ota  - n   test
 NAME   AGE
 limitrange / limits   8m

 NAME   AGE
 resourcequ  ota / quota   8m


#  kubectl   describe   limitrange / limits   resourcequ  ota /
 quota  - n  test
 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
 persistent  volumeclai  ms   0   50
 pods   0   50
 requests . storage   0   1Ti
 secrets   1   10
 services   0   20
```

```
services . loadbalanc ers 0 5
```

12.3 Edit a namespace

This topic describes how to edit a namespace.

Prerequisites

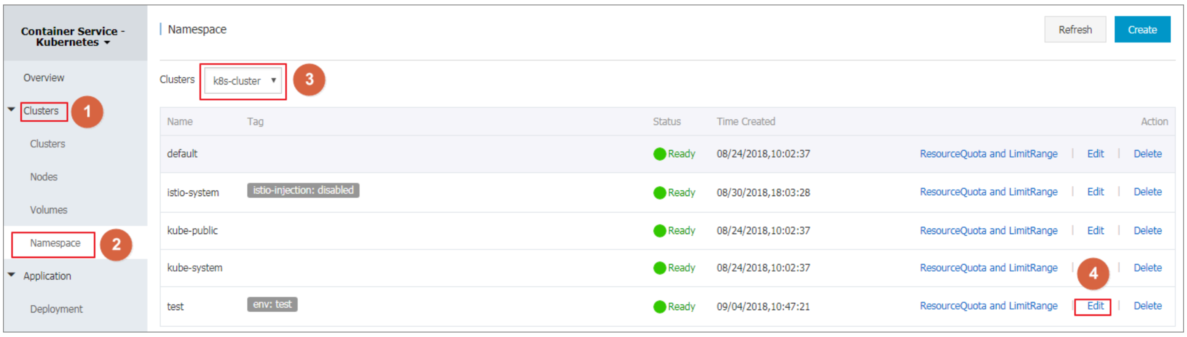
- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have created a namespace. In this topic, a namespace named test is used. For more information, see Create a namespace.

Context

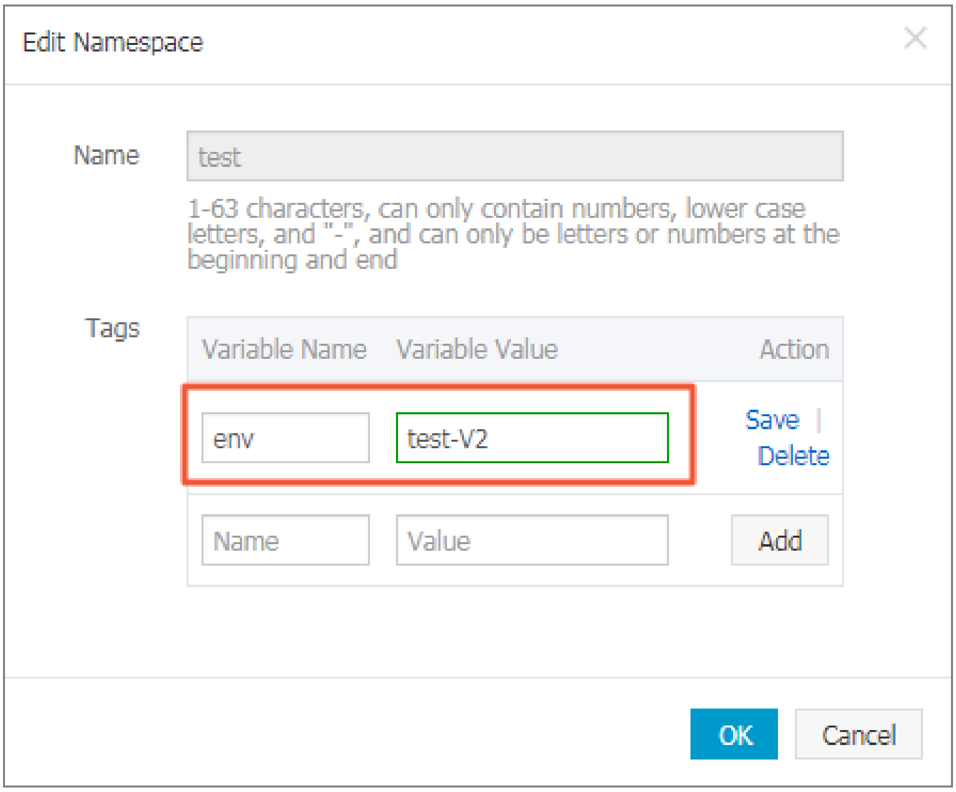
Editing a namespace means to add, modify, or delete the details of a namespace tag.

Procedure

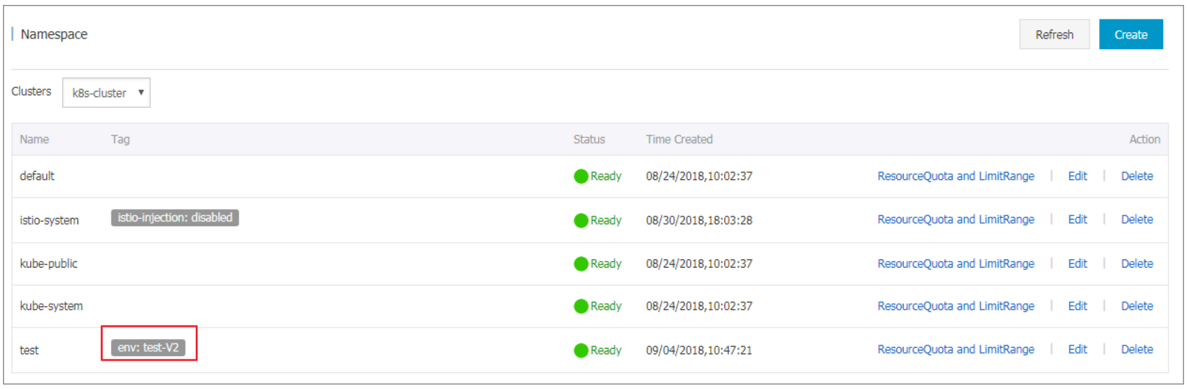
1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Namespace.
3. Select the target cluster and then click Edit on the right of the target namespace tag.



4. In the displayed dialog box, click Edit to modify the namespace tag. For example, change the tag to `env : test - V2` and click Save.



5. Click OK. The edited namespace tag is then displayed in the namespace list.



12.4 Delete a namespace

This topic describes how to delete a namespace you no longer require.

Prerequisites

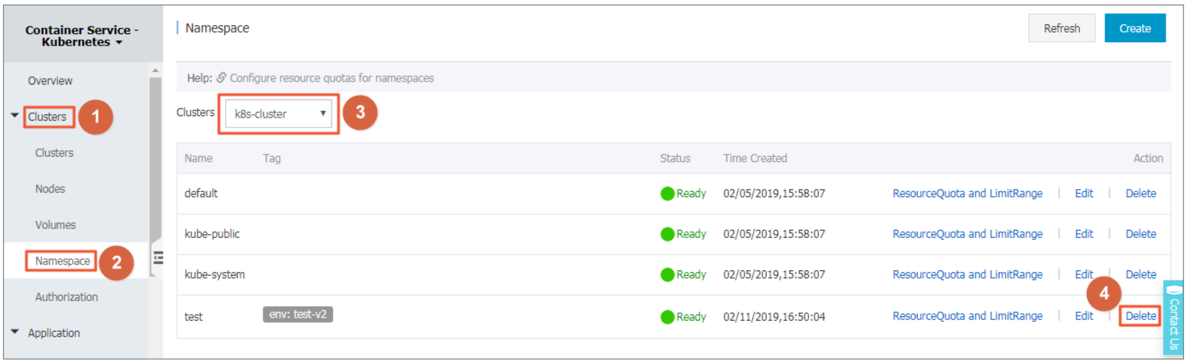
- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have created a namespace. In this topic, a namespace named test is used. For more information, see Create a namespace.

Context

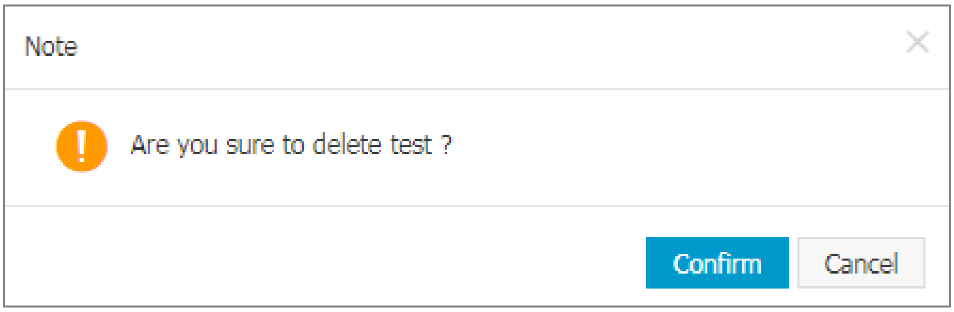
> **Note:**
> Deleting a namespace also deletes all of its resource objects. Exercise caution when performing this action.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Namespace.
3. Select the target cluster and then click Delete on the right of the cluster.



4. In the displayed dialog box, click Confirm.



5. The namespace is then deleted from the namespace list, and its resource objects are also deleted.

13 Istio management

13.1 Overview

Istio is an open platform that provides connection, protection, control and monitors microservices.

Microservices are currently being valued by more and more IT enterprises. Microservices are multiple services divided from a complicated application. Each service can be developed, deployed, and scaled. Combining the microservices and container technology simplifies the delivery of microservices and improves the liability and scalability of applications.

As microservices are extensively used, the distributed application architecture composed of microservices becomes more complicated in dimensions of operation and maintenance, debugging, and security management. Developers have to deal with greater challenges, such as service discovery, load balancing, failure recovery, metric collection and monitoring, A/B testing, gray release, blue-green release, traffic limiting, access control, and end-to-end authentication.

Istio emerged. Istio is an open platform for connecting, protecting, controlling, and monitoring microservices. It provides a simple way to create microservices networks and provides capabilities such as load balancing, inter-service authentication, and monitoring. Besides, Istio can provide the preceding functions without modifying services.

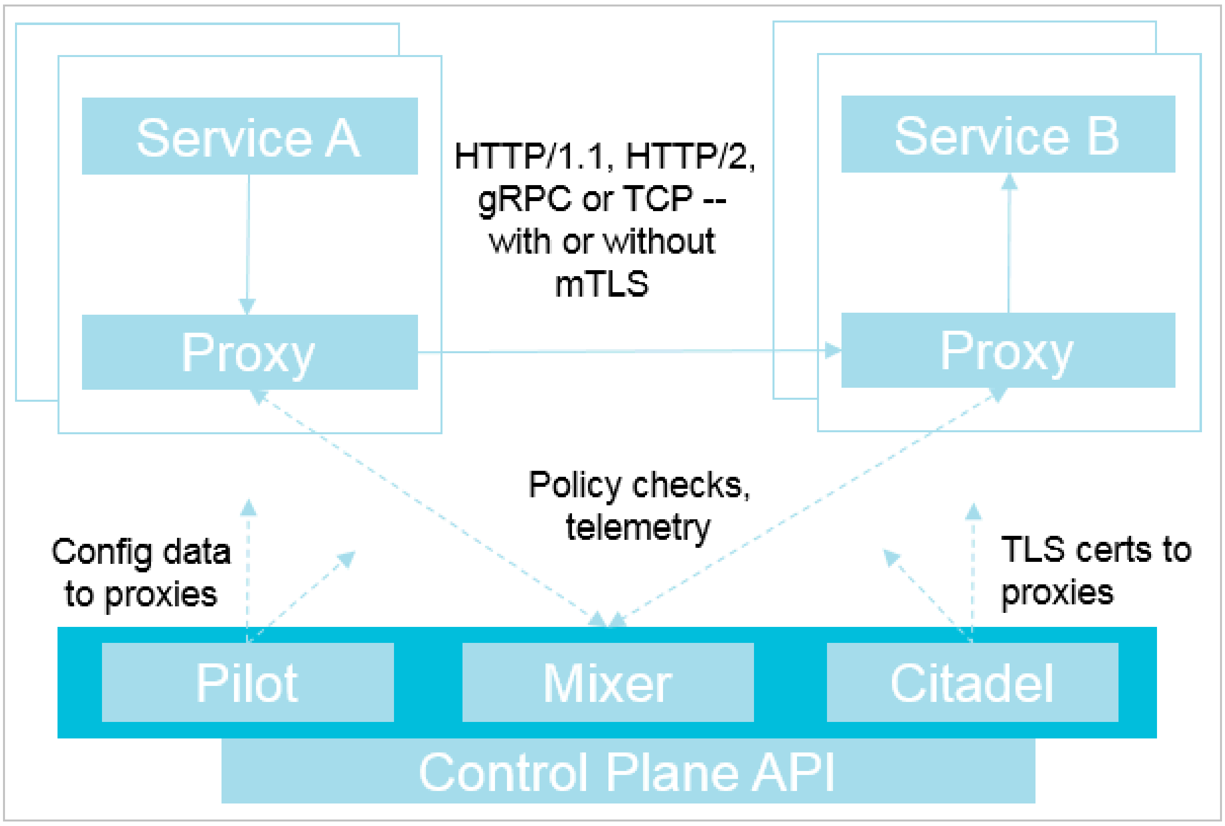
Istio provides the following functions:

- Traffic management: Controls traffic and API calls between services to enhance the system reliability.
- Authentication and security protection: Provides authentication for services in meshes, and protects the traffic of services to enhance the system security.
- Policy execution: Controls access policies between services without requiring changes to the services.
- Observability: Obtains traffic distribution and call relationships between services to quickly locate problems.

Istio architecture

Istio is logically divided into a control plane and a data plane:

- Control plane: Administration proxy (the default is Envoy) for managing traffic routing, runtime policy execution, and more
- Data plane: Consists of a series of proxys (the default is Envoy) for managing and controlling network communication between services.



Istio is composed of the following components:

- Istio Pilot: Collects and validates configurations, and propagates them to various Istio components. It extracts environment-specific implementation details from the policy execution module (Mixer) and the intelligent proxy (Envoy), providing them with an abstract representation of user services, independent of the underlying platform. In addition, traffic management rules (that is, generic Layer-4 rules and Layer-7 HTTP/gRPC routing rules) can be programmed through Pilot at runtime.
- Policy execution module (Mixer): Executes access control and usage policies across the service mesh, and collects telemetry data from the intelligent proxy (Envoy) and other services. Mixer executes policies based on the Attributes provided by the intelligent proxy (Envoy).

- Istio security module: Provides inter-service and inter-user authentication to guarantee enhanced security between services without modifying service codes. Includes three components:
  - Identification: When Istio runs on Kubernetes, it identifies the principal that runs the service according to the service account provided by container Kubernetes.
  - Key management: Provides CA automated generation, and manages keys and certificates.
  - Communication security: Provides a tunnel between the client and the server through the intelligent proxy (Envoy) to secure services.
- Intelligent proxy (Envoy): Deployed as an independent component in the same Kubernetes pod along with relevant microservice, and provides a series of attributes to the policy execution module (Mixer). The policy execution module (Mixer) uses these attributes as the basis to execute policies, and sends them to monitoring systems.

13.2 Deploy Istio on a Kubernetes cluster

This topic describes how to use the Container Service console to deploy Istio on a Kubernetes cluster that is created by Alibaba Cloud Container Service for Kubernetes (ACK).

Background information

Istio provides a service mesh that can be used to meet the requirements of the distributed application architecture that involves microservices such as applicatio n O&M, debugging, and security management. You also can use Istio for microservi ce network scenarios such as load balancing, service-to-service authentication, and monitoring.

Prerequisites

- A Kubernetes cluster is created with ACK. For more information, see Create a Kubernetes cluster.
- The version of the Kubernetes cluster is 1.10.4 or later.

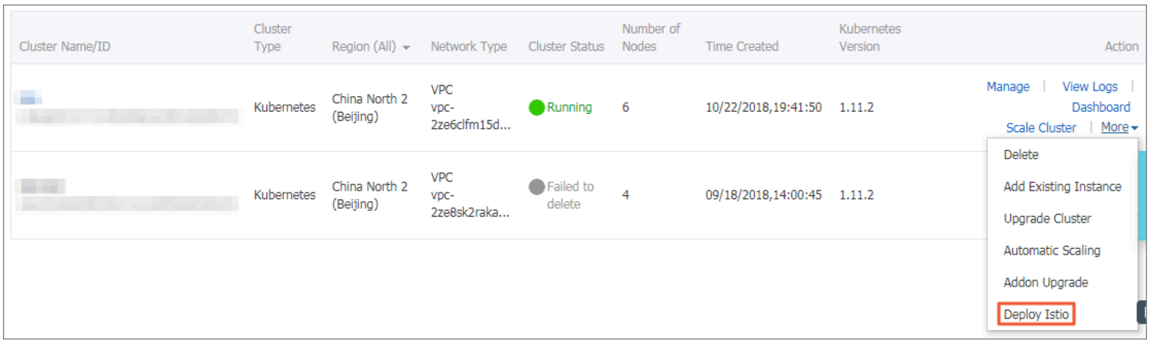
  Note:

If you are using a Kubernetes cluster of an earlier version, you must upgrade the cluster to V1.10.4 or later.

- An Alibaba Cloud account or RAM users (with sufficient permissions granted to them) are obtained to log on to Alibaba Cloud. For more information, see Grant RBAC permissions to a RAM user.
- At least three Worker nodes are set for the Kubernetes cluster.

Procedure

1. Deploy Istio on the target Kubernetes cluster.
    a. Log on to the Container Service console.
    b. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.
    c. Find the target cluster. Then, in the Action column, choose More > Deploy Istio.



    d. Set the following parameters.

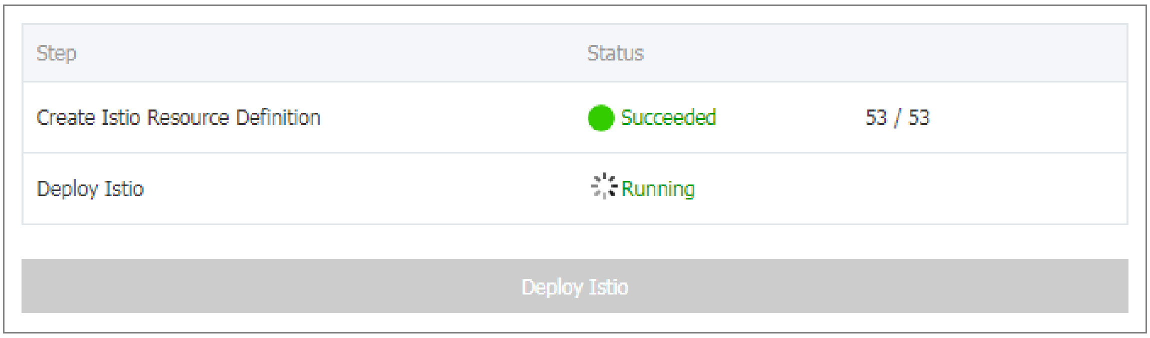
Setting	Description
Clusters	The target cluster on which Istio is deployed.
Enable Prometheus for metrics/ logs collection	Indicates whether to enable Prometheus for metrics and logs collection. By default, this is disabled.
Enable Grafana for metrics display	Indicates whether to allow Grafana to display metrics. By default, this is disabled.

Setting	Description
Enable the Kiali Visualization Service Mesh	Indicates whether to enable the Kiali visualization service mesh. By default, this is disabled. To enable this feature, set the following parameters: · Username: The default is admin. · Password: The default is admin.
Tracing Analysis Settings	Activate Tracing Analysis: Indicates whether to activate the Tracing Analysis service. To activate this service, select this radio button, and then click Activate now. You must enter an endpoint address in the format of `http :// tracing - analysis - dc - hz . aliyuncs . com /.../ api / v1 / spans .` An address of this format indicates an Internet or intranet endpoint that is used by a Zipkin client. This client is used to transmit collected data to the Tracing Analysis service, which then uses the API from v1 release. Note: If you use an intranet endpoint, you must ensure that your Kubernetes cluster and the Tracing Analysis instance are in the same region to maintain stable network performance.
Pilot Settings	The trace sampling percentage. The value range is from 0 to 100. The default value is 1.

Setting	Description
Control Egress Traffic	· Permitted Addresses for External Access: Indicates the range of IP addresses that can be used to directly access services in the Istio service mesh. By default, this field is left unspecified. Use commas (,) to separate multiple IP address ranges. · Blocked Addresses for External Access: Indicates the range of IP addresses that are blocked against external accesses. By default, this IP address range contains the cluster pod CIDR block and service CIDR block. Use commas (,) to separate multiple IP address ranges. · ALL: Select this check box to block all the IP addresses used to access the Internet. Note: If the settings of these two parameters conflict with each other, the Permitted Addresses for External Access prevails.

e. Click Deploy Istio.

At the bottom of the deployment page, you can view the deployment progress and status in real time.

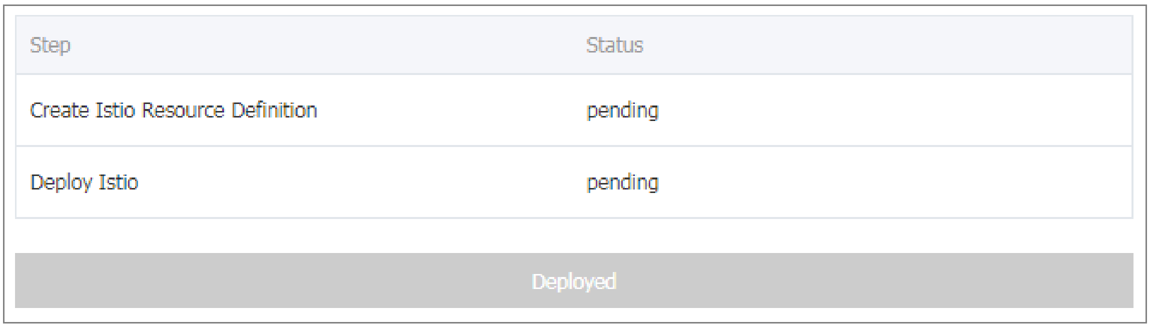


Verify the results

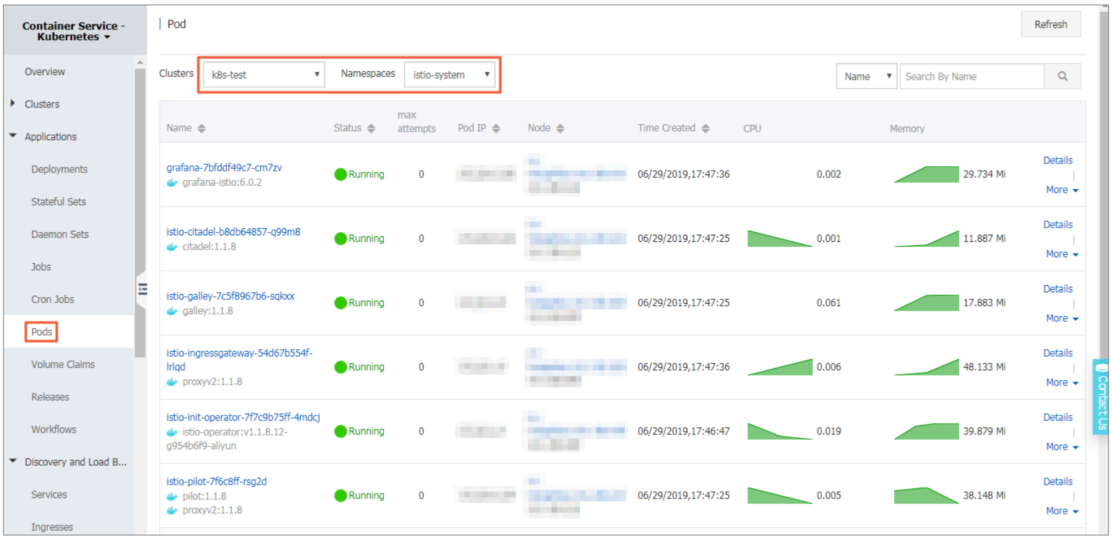
To verify that Istio is deployed on the target Kubernetes cluster, perform the following operations:

- View the status of this deployment.

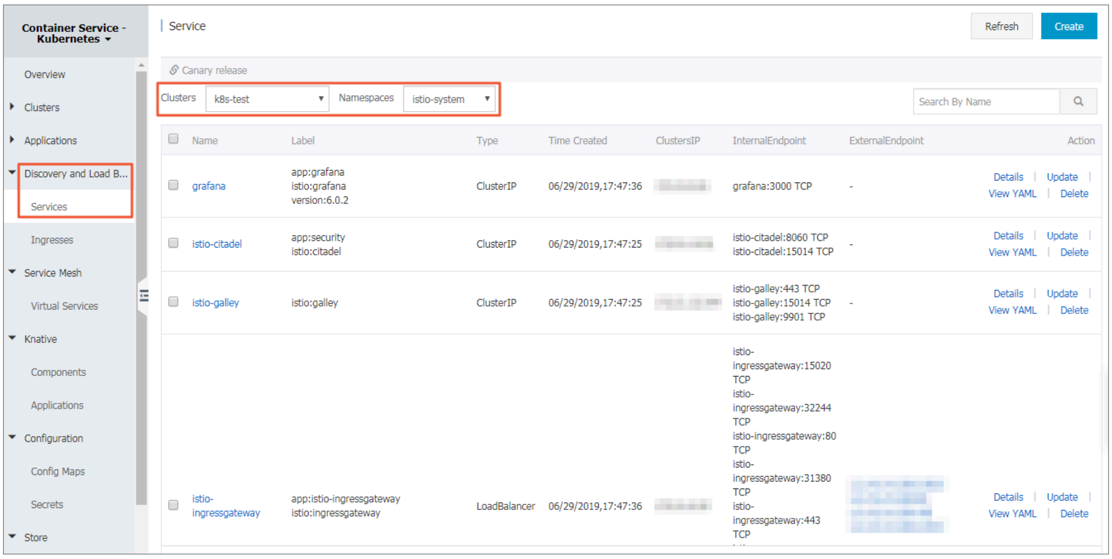
At the bottom of the Deploy Istio page, verify that Deploy Istio is changed to Deployed.



- View the pods on which Istio is deployed.
  a. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Pods.
  b. Select the cluster and namespace where Istio is deployed.



- View the services on which Istio is deployed.
  a. In the left-side navigation pane under Container Service-Kubernetes, choose Discovery and Load Balancing > Services.
  b. Select the cluster and namespace where Istio is deployed.



2. Modify the Istio ingress gateway.

> **Note:**
> If you deploy Istio 1.1.4 or later on a Kubernetes cluster, an ingress gateway can be automatically created. Therefore, we recommend that you upgrade Istio to the latest version.

a. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Releases.

b. Click the Helm tab.

c. In the Action column, click Update.

The displayed page shows the settings as follows:

```
gateways :
  enabled :  true
  ingress :
   - enabled :  true
     gatewayNam  e :  ingressgat  eway
     maxReplica  s :  5
     minReplica  s :  1
     ports :
      - name :  status - port
        port :  15020
        targetPort :  15020
      - name :  http2
        nodePort :  31380
        port :  80
        targetPort :  80
      - name :  https
        nodePort :  31390
        port :  443
        targetPort :  0
      - name :  tls
        port :  15443
        targetPort :  15443
```

```
        replicaCou  nt :  1
        serviceTyp  e :  LoadBalanc  er
    k8singress : {}
```

d. Modify settings of the ingress gateway, and then click Update.

> Note:
>
> - `replicaCou nt` : Set the number of replicas.
> - `ports` : Set ports.
> - `serviceTyp e` : Set the service type. Valid value: `LoadBalanc er` | `ClusterIP` | `NodePort` .
> - `serviceAnn otations` : (If you set `serviceTyp e` as `LoadBalanc er` , this parameter is available.) Set whether to use an Internet or intranet SLB instance, and whether to use an existing SLB instance or create a new SLB instance. For more information, see Server Load Balancer.

13.3 Update Istio

You can modify the deployed Istio through updates.

Prerequisites

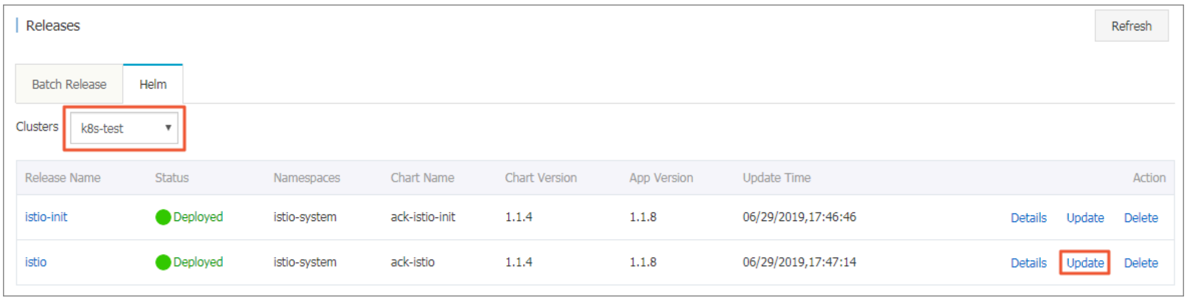
- You have created an Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have created an Istio. For more information, see Deploy Istio on a Kubernetes cluster.

Procedure

1. Log on to the Container Service console.
2. Under the Kubernetes menu, click Application > Helm in the left-side navigation pane.
3. Select a cluster, select the Istio to be updated, and click Update in the action column.

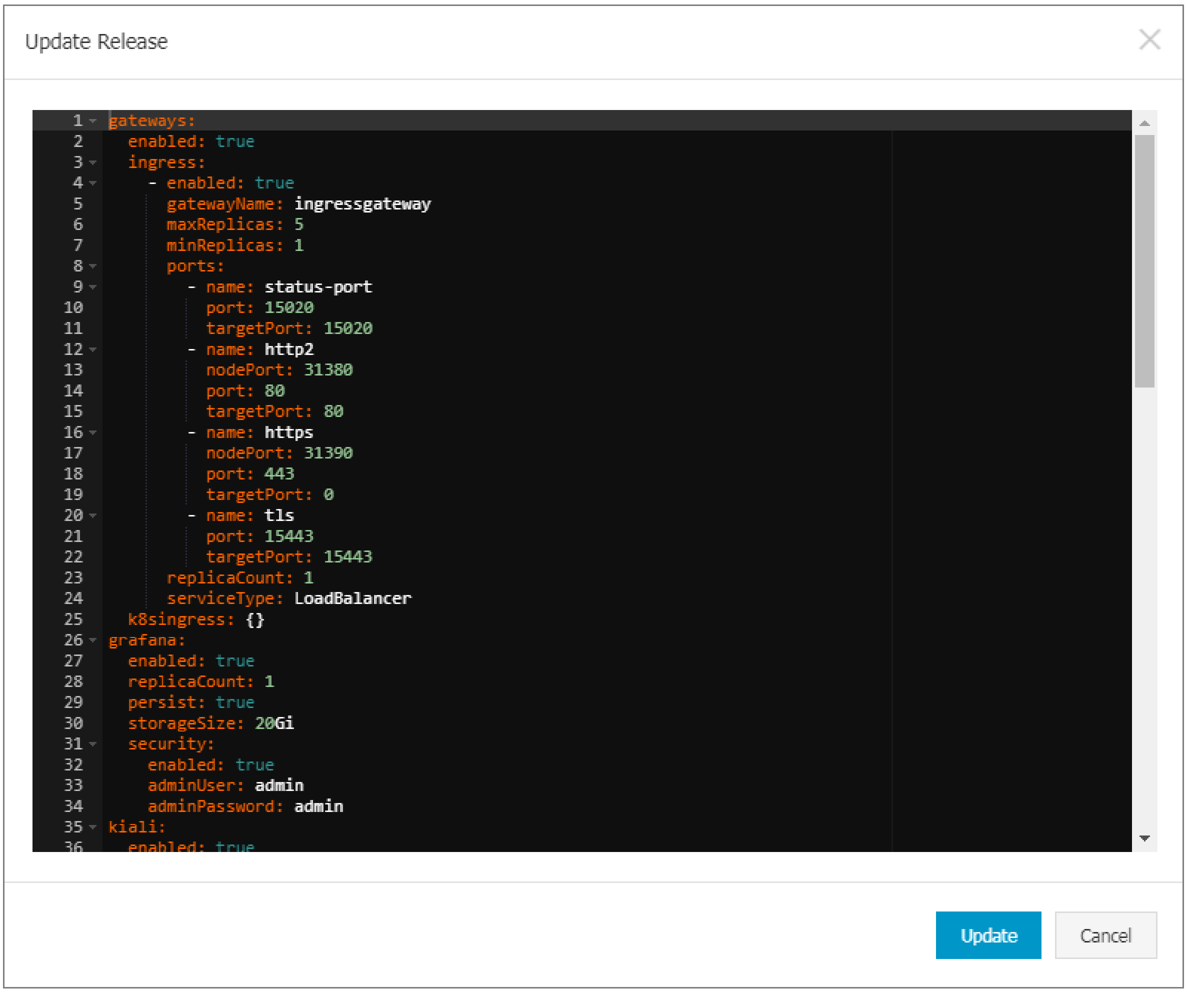
> Note:
>
> - The release name of the Istio that is deployed through the cluster interface is istio. Configurations to be updated are the same as the options configured in deployment.

- The release name of the Istio that is deployed through the application catalog is the name specified when you create the Istio. Configurations to be updated are the same as the options configured in deployment.



4. In the displayed dialog box, modify parameters of the Istio, and then click Update.

   In this example, update the Istio that is deployed through the cluster interface:



Result

You can view updated content in two ways:

- After you complete the update, the page automatically jumps to theRelease List page. On the Resource tab, you can view updated content.

- Under the Kubernetes menu, click Application > Pods, and select the target cluster and namespace to view updating results.

13.4 Delete Istio

This topic describes how to use the Container Service console to delete the deployed Istio.

Prerequisites

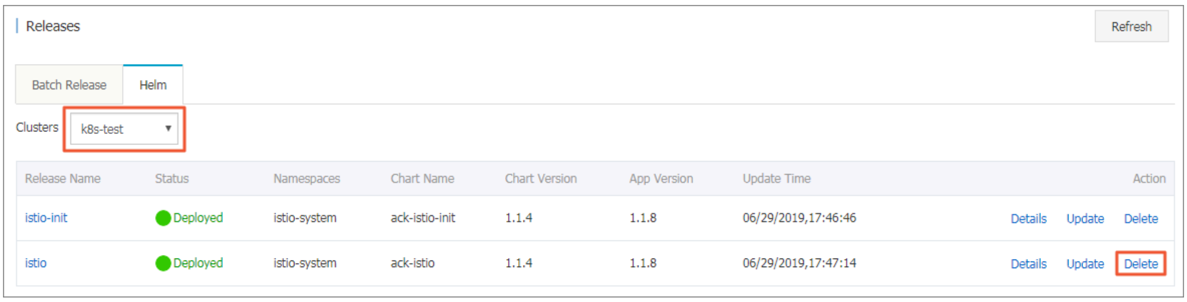
- A Kubernetes cluster is created with ACK. For more information, see Create a Kubernetes cluster.
- Istio is deployed on the Kubernetes cluster. For more information, see Deploy Istio on a Kubernetes cluster.
- `istio - init` on the Helm tab page of the target cluster is in the running status.

> **Note:**
> You must first delete the deployed Istio before you can delete `istio - init` .

- The Master node of the Kubernetes cluster is accessible. For more information, see Connect to a Kubernetes cluster by using kubectl.

Procedure

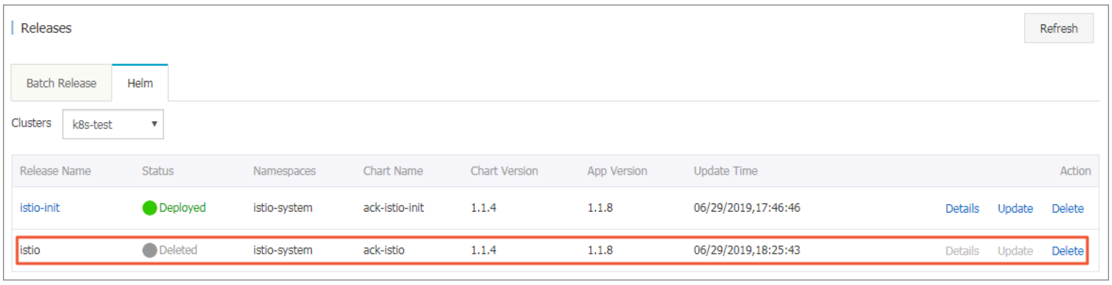
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Release.
3. Click the Helm tab and select the target cluster to find the deployed Istio. Then, in the Action column, click Delete.



4. In the displayed dialog box, click OK.

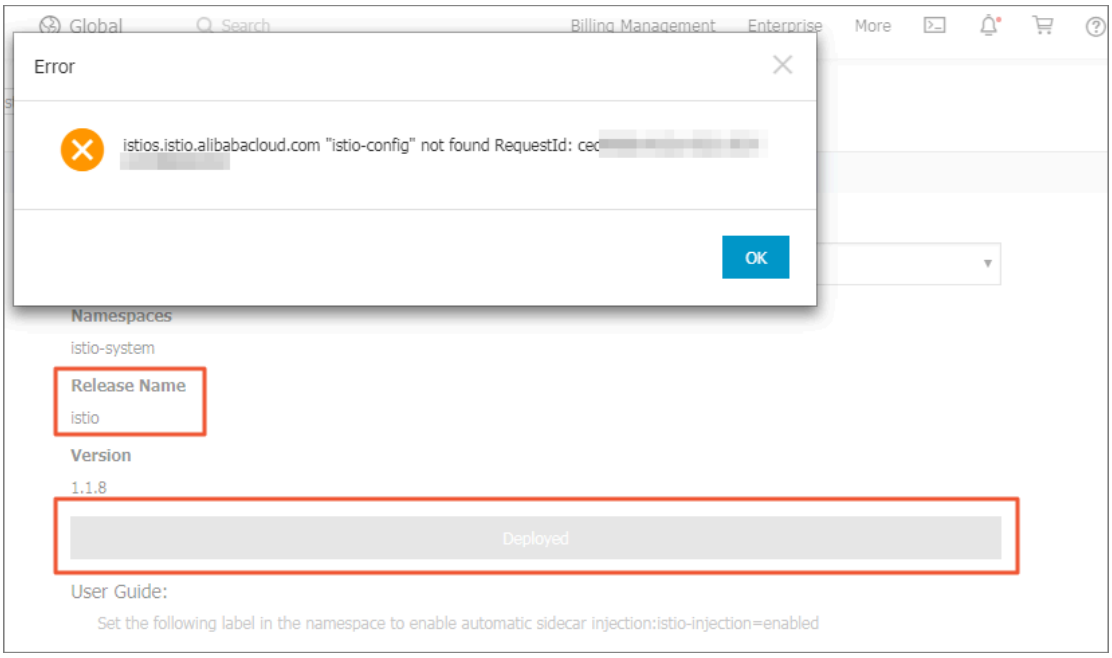
The displayed dialog box contains a check box named Purge. By default, it is selected. We recommend that you retain the default setting.

- If you do not select this check box, the following settings apply:
  - The release record of `istio` is retained.



  - Istio cannot be deployed again on the target cluster.

    Specifically, if you redeploy Istio on the target cluster, the system prompts an error message and also shows that Istio is deployed.



- If you retain the check box selected, all release records are deleted. This means that Istio can be deployed again on the target cluster.

5. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Pods.
6. Select the target cluster and the `istio - system` namespace to verify that pods related to Istio are deleted.

> Note:
> If the `istio - config` resource in the `istio - system` namespace cannot be deleted, see What do I do if a custom resource is retained after I deleted Istio.

7. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Releases.
8. Click the Helm tab and select the target cluster to find `istio - init`. Then, in the Action column, click Delete.

   This action deletes the operator pods that used to deploy Istio.

Result

1. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Pods.
2. Select the target cluster and the `istio - system` namespace to verify that pods related to Istio are deleted.

13.5 Upgrade Istio components

This topic describes how to upgrade Istio components.

Background information

- The Istio upgrade may install new binaries, and change configurations and API schemas.
- The upgrade process may cause service downtime.
- To minimize downtime, use multiple replicas to ensure that your Istio control plane components and your applications remain highly available.
- This upgrade may recreate the existing SLB instance of your services. Therefore, we recommend that you enable the SLB protection feature to prevent the existing SLB instance from being deleted.

Note:
In the following example, assume that the Istio components are installed and upgraded in the istio-system namespace.

Procedure

To complete the upgrade process, you need to upgrade CRD files, the control plane, and the data plane sidecar.

Note:
The upgrade may cause downtime. To reduce the downtime, you must use multiple replicas to ensure the Istio control plane and your applications are highly available.

Upgrade CRD files

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Releases. Then, click the Helm tab.

   Note:

   - If istio-init is displayed and it is earlier than V1.1.4, click Delete.
   - If istio-init is displayed and it is V1.1.4 or later, click Update. Then, in the displayed dialog box, click Update.

3. In the left-side navigation pane under Container Service-Kubernetes, choose Marketplace > App Catalog. Then, click ack-istio-init.
4. In the Deploy area, select the target cluster. Then, click DEPLOY.

   Note:
   By default, the istio-system namespace is selected, and the release name is set to istio-init.
5. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Pods. Then, click the podistio-init-operator-xxxxxxx-xxxxx.
6. Click the Logs tab and check the pod logs to determine whether the pod has executed the upgrade.

   Note:
   - If the pod logs indicate that the pod has executed the upgrade, move on to the next step.
   - If the pod logs indicate that the pod has not executed the upgrade, return to the Pods page to delete this pod.
7. In the left-side navigation pane under Container Service-Kubernetes, choose Application > Release. Then, click the Helm tab to verify that the Istio component is upgraded.

Upgrade the control plane

The Istio control plane components include the Citadel, Pilot, Policy, Telemetry, and Sidecar injector.

1. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Releases.
2. Select the target cluster, select the target Release Name, and click Upgrade in the Action column.
3. In the configurations of the deployed Istio, specify the version number of the Istio to be deployed.

   Note:
   On this page, you can also modify other parameters.

   ```
   global :
   ```

```
        tag : < enter   the   version   number >
```

4. Click Update.

Upgrade the data plane sidecar

Note that after you upgrade the control plane, the applications that have already run Istio will still use the sidecar of an earlier version. To upgrade the sidecar, you need to re-inject it.

Automatic sidecar injection

If you use automatic sidecar injection, you can upgrade the sidecar by performing a rolling update for all pods. Then, the sidecar of the new version will be automatically re-injected.

You can use the following script to trigger the rolling update by patching the termination grace period.

```
NAMESPACE =$ 1
     DEPLOYMENT  _LIST =$( kubectl  - n  $ NAMESPACE   get
deployment  - o   jsonpath ='{. items [*]. metadata . name }')
     echo  " Refreshing   pods   in   all   Deployment  s : $
DEPLOYMENT  _LIST "
     for   deployment  _name   in  $ DEPLOYMENT  _LIST  ;  do
    # echo  " get   TERMINATIO  N_GRACE_PE  RIOD_SECON  DS   from
deployment : $ deployment  _name "
     TERMINATIO  N_GRACE_PE  RIOD_SECON  DS =$( kubectl  - n  $
NAMESPACE   get   deployment  "$ deployment  _name " - o   jsonpath
='{. spec . template . spec . terminatio  nGracePeri  odSeconds }')
     if  [ "$ TERMINATIO  N_GRACE_PE  RIOD_SECON  DS " - eq   30  ];
then
     TERMINATIO  N_GRACE_PE  RIOD_SECON  DS =' 31 '
     else
     TERMINATIO  N_GRACE_PE  RIOD_SECON  DS =' 30 '
     fi
     patch_stri  ng ="{\" spec \":{\" template \":{\" spec \":{\"
terminatio  nGracePeri  odSeconds \":$ TERMINATIO  N_GRACE_PE
RIOD_SECON  DS }}}}"
    # echo  $ patch_stri  ng
     kubectl  - n  $ NAMESPACE   patch   deployment  $ deployment
_name  - p  $ patch_stri  ng
     done
     echo  " done ."
```

Manual sidecar injection

Run the following command to manually upgrade the sidecar:

```
kubectl  apply  - f  <( istioctl  kube - inject  - f  $ ORIGINAL_D
EPLOYMENT_  YAML )
```

If the sidecar was previously injected with some customized injection configuration files, run the following command to manually upgrade the sidecar:

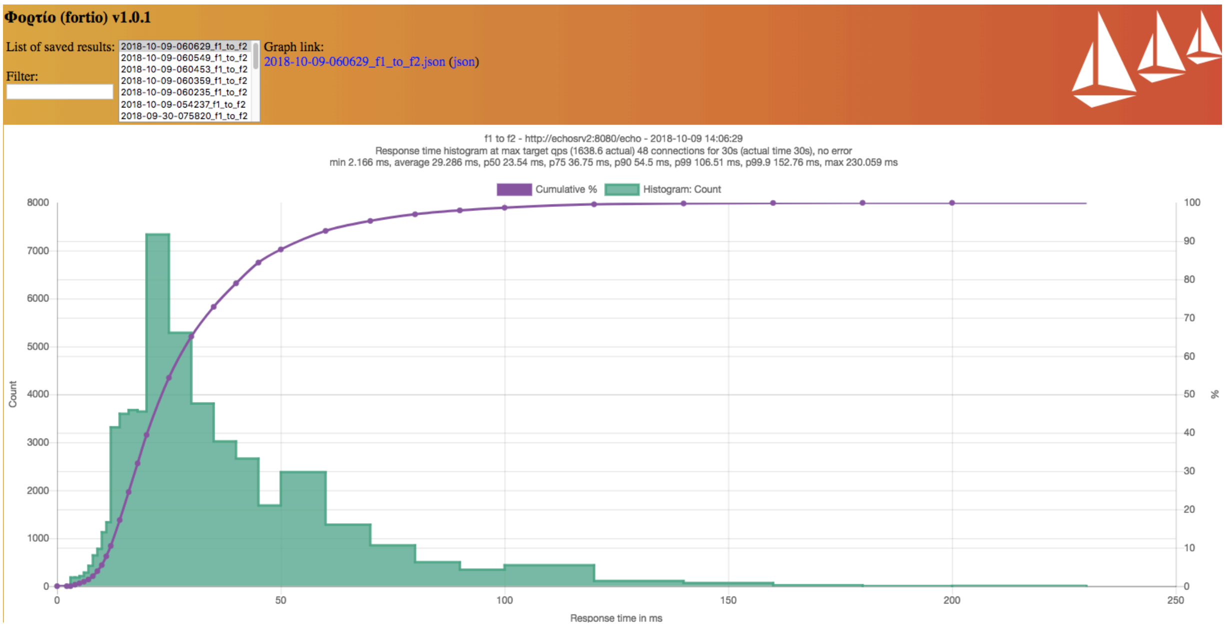
```
kubectl  apply  - f  <( istioctl  kube - inject  -- injectConf
igFile  inject - config . yaml  -- filename  $ ORIGINAL_D
EPLOYMENT_  YAML )
```

Impacts caused by the Istio upgrade

Impacts caused by the CRD file upgrade

The upgrade process does not impact the calls between services within the cluster or the calls from the gateway to services.

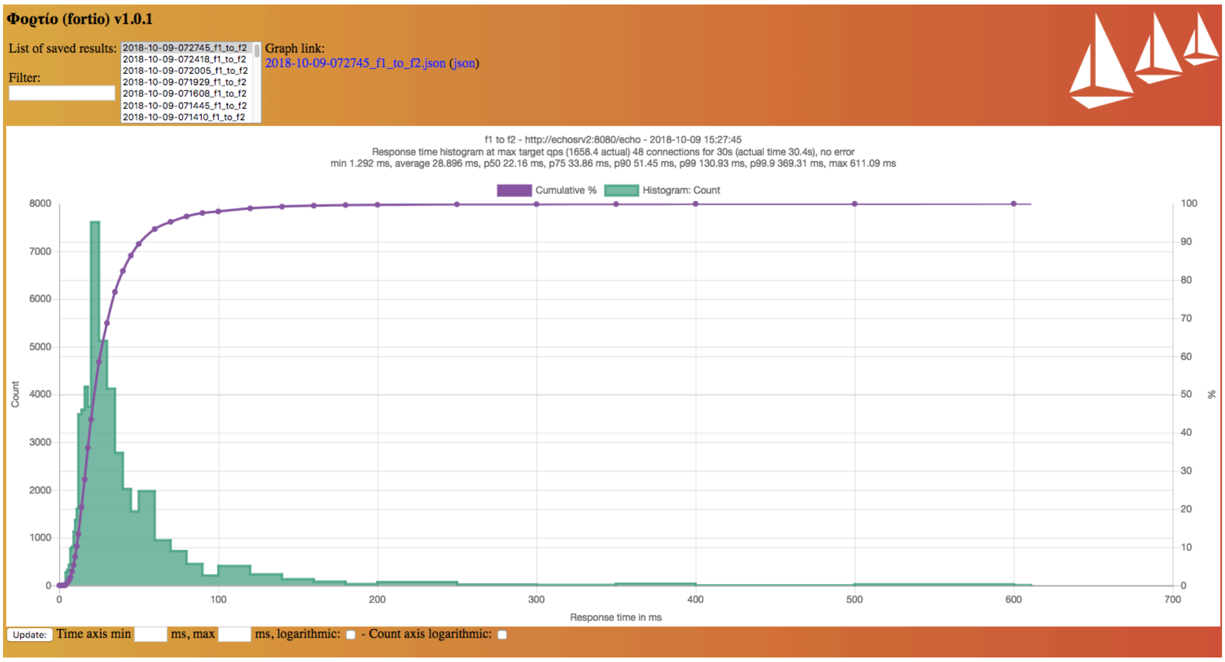
Calls between services within the cluster.



Impacts caused by the control plane upgrade

If HA is enabled, that is, the replicas of Pilot is 2, the HPA setting of `istio - pilot / istio - policy / istio - telemetry` is `minReplica s : 2`.

If you have changed the Istio version multiple times by upgrading or rolling back the component version, testing results will indicate that the QPS of calls between services remains unchanged and the calls proceed normally.



Impacts caused by the control plane sidecar upgrade

No obvious change occurs to both the QPS of the calls between services within the cluster and the QPS of the calls from the gateway to services. But these calls will terminate temporarily. We recommend that you use multiple replicas to upgrade the sidecar to reduce the impacts.

13.6 Deploy Istio on Kubernetes clusters across multiple regions

This topic describes how to deploy Istio on Kubernetes clusters created with Alibaba Cloud Container Service for Kubernetes (ACK) across multiple regions. Istio enables you to manage all of the traffic destined for these clusters. For example, if a Kubernetes cluster near to you fails or is overloaded, Istio deployed with such a method can seamlessly redirect the traffic destined for the cluster to another available cluster.

Prerequisites

- A Kubernetes cluster is created with ACK. For more information, see Create a Kubernetes cluster.
- An Alibaba Cloud account is obtained, or a RAM user that is granted with required permissions is obtained.

> **Note:**

For example, you can set the `cluster - admin` role for a RAM user to grant corresponding permissions to the RAM user. For more information, see Grant RBAC permissions to a RAM user.

- If you use a Flat network or VPN to deploy Istio on multiple Kubernetes clusters, the clusters must be located in the same VPC.
- If you do not use a Flat network or VPN to deploy Istio on multiple Kubernetes clusters, the clusters can be located in different VPCs. In this case, the VPCs must be connected by using Cloud Enterprise Network (CEN) or Express Connect.
- A network is planed to ensure that the pod CIDR blocks, service CIDR blocks, and VPC CIDR blocks of involved Kubernetes clusters do not overlap withe each other.

Note:
In this topic, the Flat network or VPN is used to deploy Istio on multiple Kubernetes clusters.

- The Kubernetes cluster version is 1.10.4 or later. Any earlier version does not support Istio. Therefore, you must upgrade any earlier version to 1.10.4 or later.
- At least three Worker nodes are set for the Kubernetes cluster.

Procedure

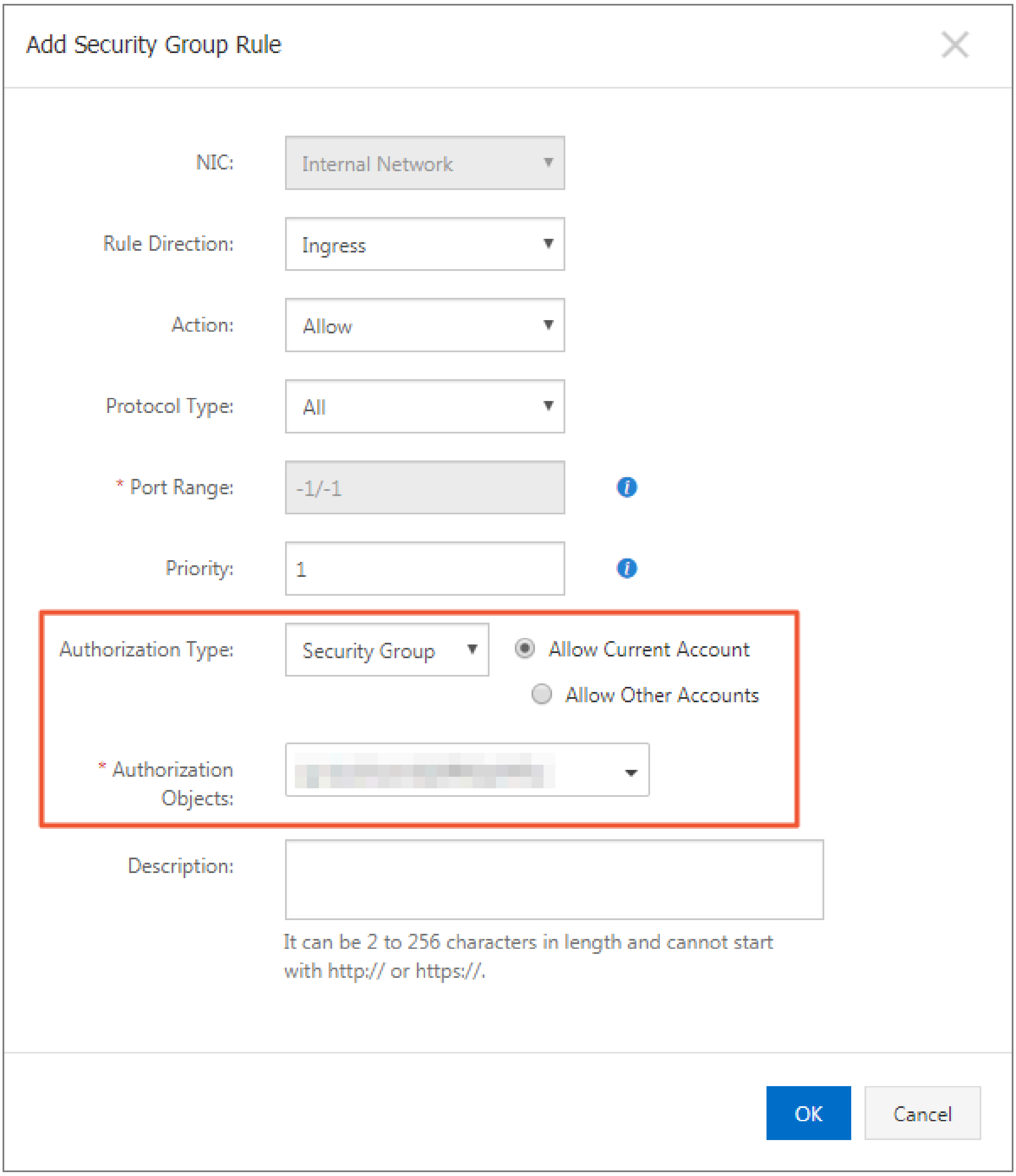
Step 1: Set the network for the target Kubernetes clusters

- To enable Kubernetes clusters located in the same VPC to communicate with each other, add security group rules.

Note:

- By default, Kubernetes clusters located in the same VPC that belong to different security groups, which in turn causes them to be unable to communicate with each other.
- To allow multiple security groups in the VPC to communicate with each other , you must manually configure rules to allow them to communicate with each other.

- For more information, see Add security group rules.



- To enable Kubernetes clusters located in the different VPCs to communicate with each other, complete the following settings:
  - Connect the VPCs by using a CEN instance, Express Connect, or a VPN gateway.

    Note:

    - We recommend that you use a CEN instance because configuring this instance type is relatively easy and because this instance type can automatically distribute and learn routes. For more information, see Connect VPCs.

- When a Kubernetes cluster is created, the default CIDR block 192.168.0.0/16 is used to create the VPC for the Kubernetes cluster. Therefore, you must set different CIDR blocks to create the VPCs for Kubernetes clusters.

- Add security group rules to the security groups to which the Kubernetes clusters belong.

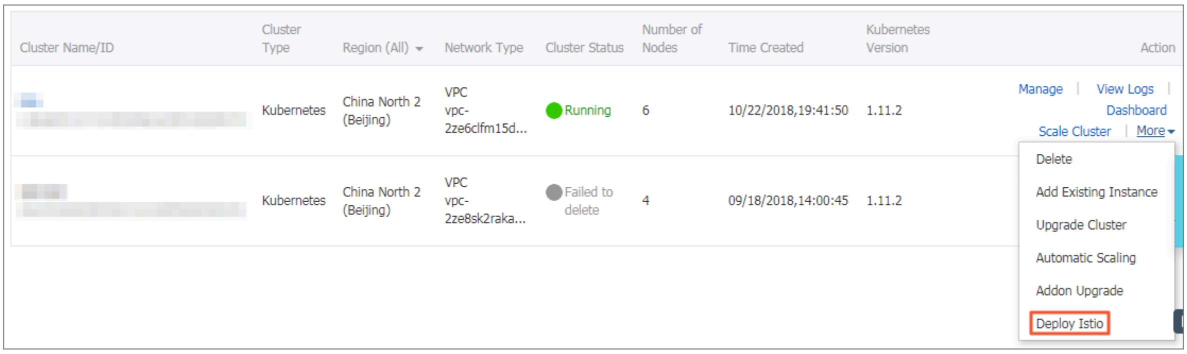
Note:

For more information, see Add security group rules.

- Add a security group rule to the security group on the data plane to allow access from the CIDR block where the control plane belongs.
- Add a security group rule to the security group on the control plane to allow access from the CIDR block where the data plan belongs.

Step 2: Deploy Istio through the Clusters page

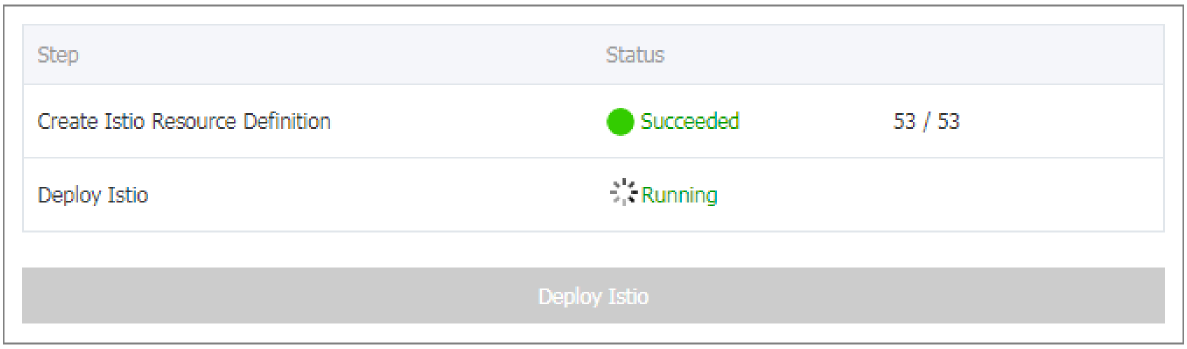
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.
3. Find the target cluster. Then, in the Action column, choose More > Deploy Istio.



4. To deploy Istio on multiple Kubernetes clusters, set the following two parameters in addition to the parameters required to deploy Istio on a Kubernetes cluster:
    - Enable locality based service routing: Select this check box to route requests to the Kubernetes cluster located in the region nearest to the region where the request are sent from. By default, this check box is not selected.
    - Multiple clusters with a Single Control Plane:
        - Disable: Do not enable the multi-cluster mode.
        - Use Flat Networks or VPNs: Use Flat networks or VPNs to enable pods of different Kubernetes cluster to communicate with each other.
        - No Flat Network or VPN: Only use gateways to enable mutual communication for different Kubernetes clusters.

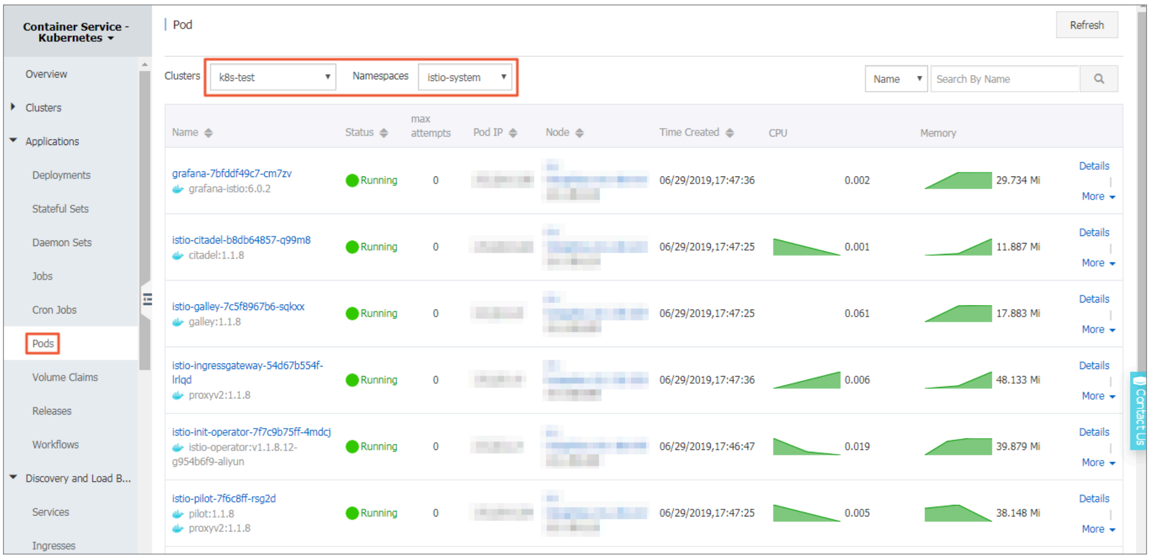
5. Click Deploy Istio.

   The page shows the deployment progress and status in real time.



   Verify the result

   a. In the left-side navigation pane, choose Application > Pods.
   b. Select the target cluster and namespace to view the pods on which Istio is deployed.



Step 3: Manage the ingress gateway of Istio

Run the `kubectl get service -n istio-system -l istio=ingressgateway` command to obtain the Internet IP address of the ingress gateway.

> **Note:**
> The Istio deployed in the preceding steps contains an ingress gateway that uses an Internet IP address to provide load balancing services. Applications that run in the Kubernetes clusters can be accessed through this IP address.

```
✔  kubectl get service -n istio-system -l istio=ingressgateway
NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)
                       AGE
istio-ingressgateway   LoadBalancer   [illegible]      [illegible]      15020:31040/TCP,80:31380/TCP,443:31390/TCP,1
5443:30221/TCP    6d22h
```

Step 4: Add the cluster where the data plane runs into Istio

1. In the Kubernetes cluster where the data plane of Istio runs, run the following command to generate a kubeconfig file:

```
kubectl   create   namespace   istio - system
       kubectl   apply  - f   http :// istio . oss - cn - hangzhou
. aliyuncs . com / istio - operator / rbac . yml
       wget   http :// istio . oss - cn - hangzhou . aliyuncs . com
/ istio - operator / generate - kubeconfig . sh
       chmod  + x   generate - kubeconfig . sh
      ./ generate - kubeconfig . sh  ${ CLUSTER_NA  ME }
```

Note:

The *${CLUSTER_NAME}* must be unique in the service mesh.

2. In the Kubernetes cluster where the control plane of Istio runs, run the following command:

```
wget   http :// istio . oss - cn - hangzhou . aliyuncs . com /
istio - operator / create - secret . sh
       chmod  + x   create - secret . sh
      ./ create - secret . sh  ${ CLUSTER_NA  ME }
```

3. In the Kubernetes cluster where the control plane of Istio runs, create a file *${CLUSTER_NAME}.yaml* and copy the following code into the file:

```
apiVersion :  istio . alibabaclo  ud . com / v1beta1
       kind :  RemoteIsti  o
       metadata :
       name : ${ CLUSTER_NA  ME }
       namespace :  istio - system
       spec :
       autoInject  ionNamespa  ces :
      -  default
       dockerImag  e :
       region :  cn - beijing
       gateways :
       k8singress : {}
       hub :  registry . cn - beijing . aliyuncs . com / aliacs -
app - catalog
       includeIPR  anges : '*'
       proxy : {}
       security : {}
       sidecarInj  ector :
       enabled :  true
```

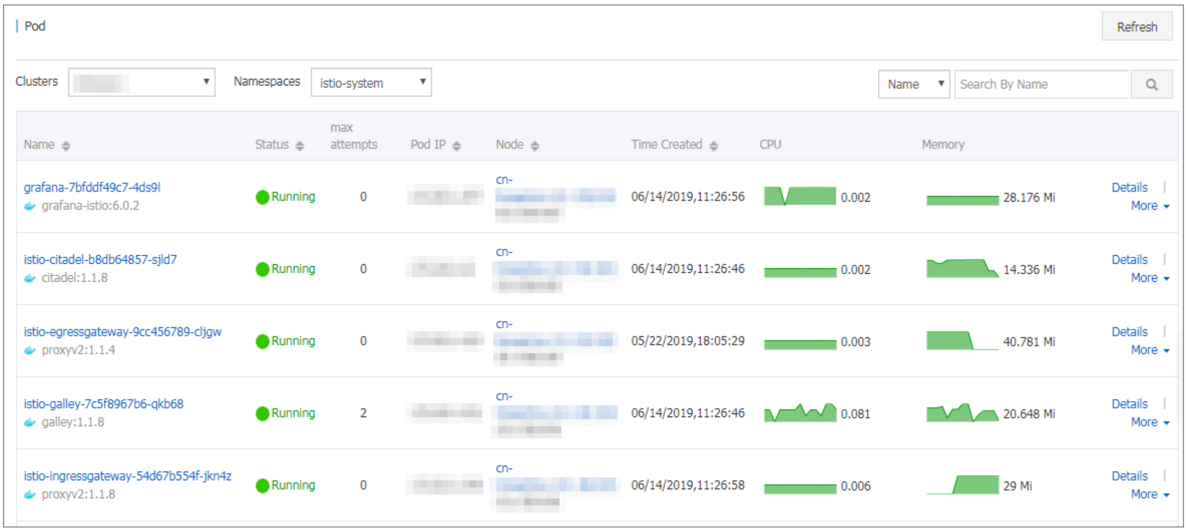
```
        replicaCount : 1
```

4. Run the `kubectl apply -n istio-system -f ${CLUSTER_NAME}.yaml` command.

Verify the result

To check if Istio is deployed in the remote Kubernetes cluster, follow these steps:

1. Log on to the remote Kubernetes cluster.
2. In the left-side navigation pane, choose Application > Pod.
3. Select the target cluster and namespace to view the pods on which Istio is deployed.



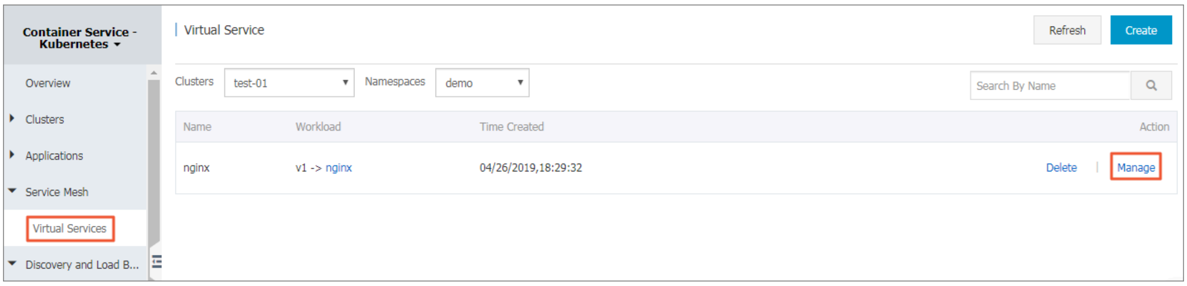
13.7 Manage traffic

This topic describes how to manage traffic by using Istio. Specifically, you can use Istio to set corresponding parameters for a load balancing algorithm, session affinity, connection pool, circuit breaker, or faulty injection.

Before you begin

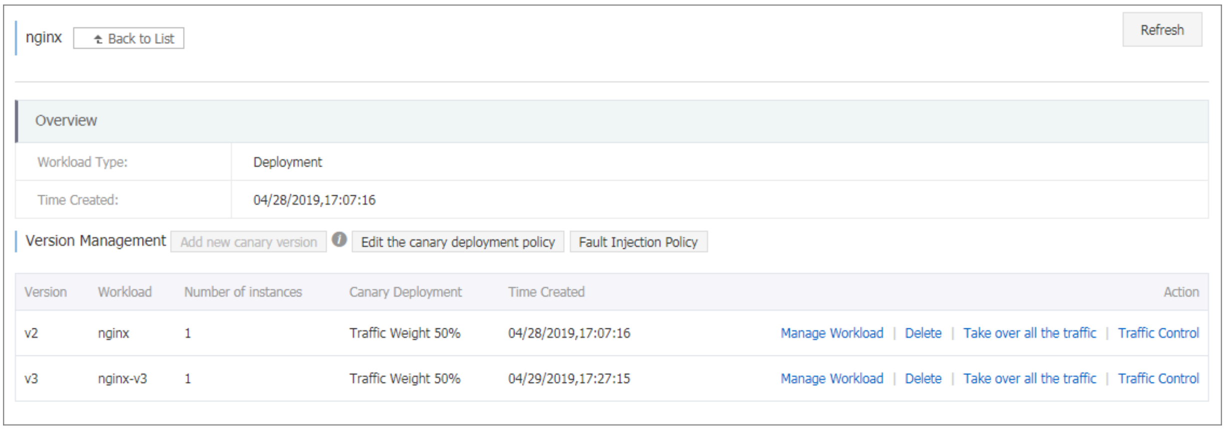
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Service Mesh > Virtual Services.

3. On the right of the target virtual service, click Manage. Then, you can manage traffic related to the service according to your needs.



Control traffic

Find the required version of the target virtual service, and then click Traffic Control.



- Set load balancing.

  Istio provides two methods to set load balancing: Load Balancer Algorithm and Session Affinity.

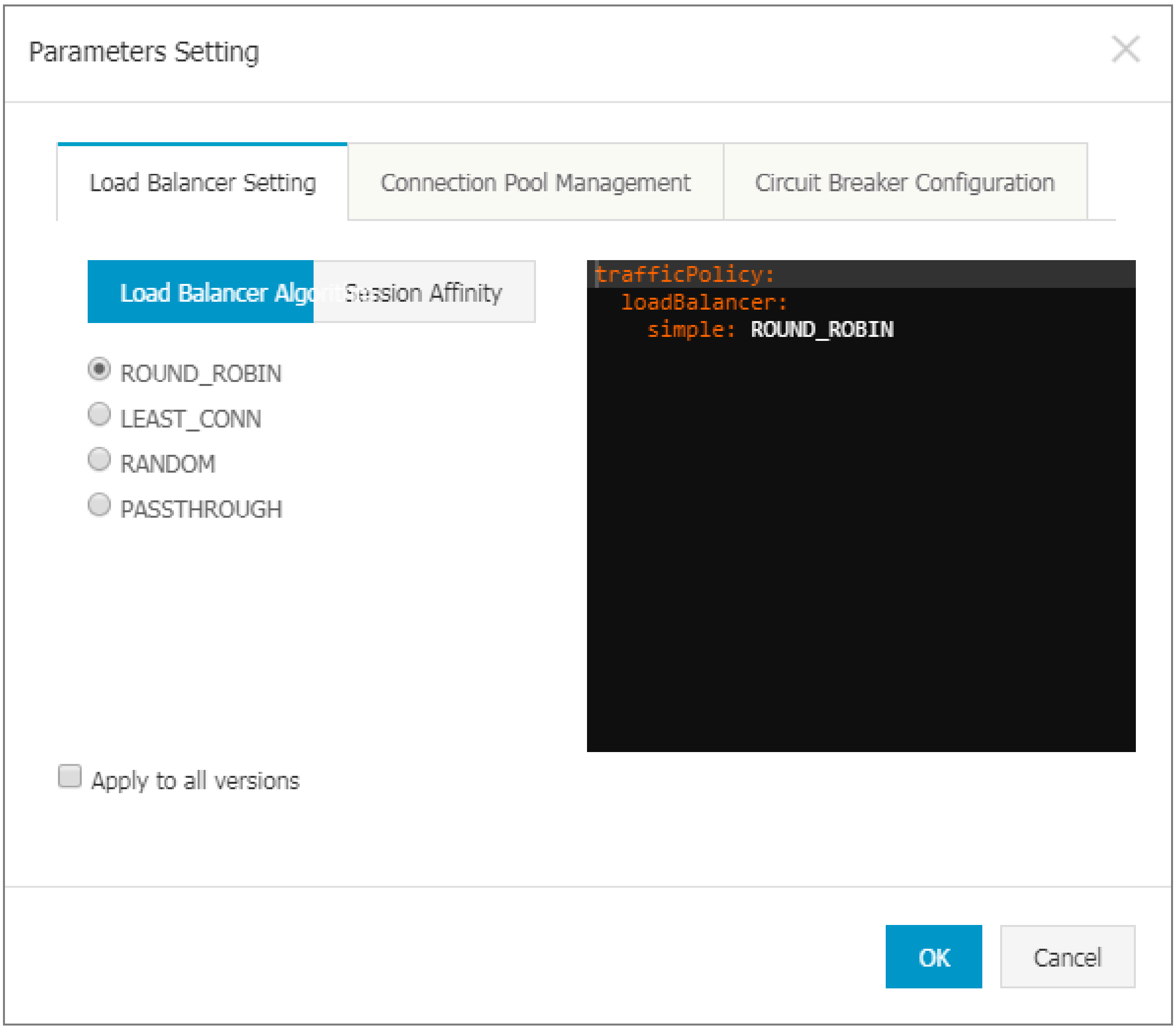
  Note:

These two methods are mutually exclusive.

Select a load balancing algorithm according to your needs.

- ROUND_ROBIN: Evenly distributes loads across the endpoints in the load balancing pool. This is the default algorithm used by the Istio proxy.
- LEAST_CONN: Selects two healthy hosts randomly and uses the host with fewer requests to provide service.
- RANDOM: Selects one healthy host randomly and evenly distributes loads on the endpoints in the load balancing pool. In the case that you have not set health checks, this is more efficient than ROUND_ROBIN.
- PASSTHROUGH: Forwards requests directly to the IP address specified by the client. For security purposes, we recommend that you exercise caution before implementing this algorithm.

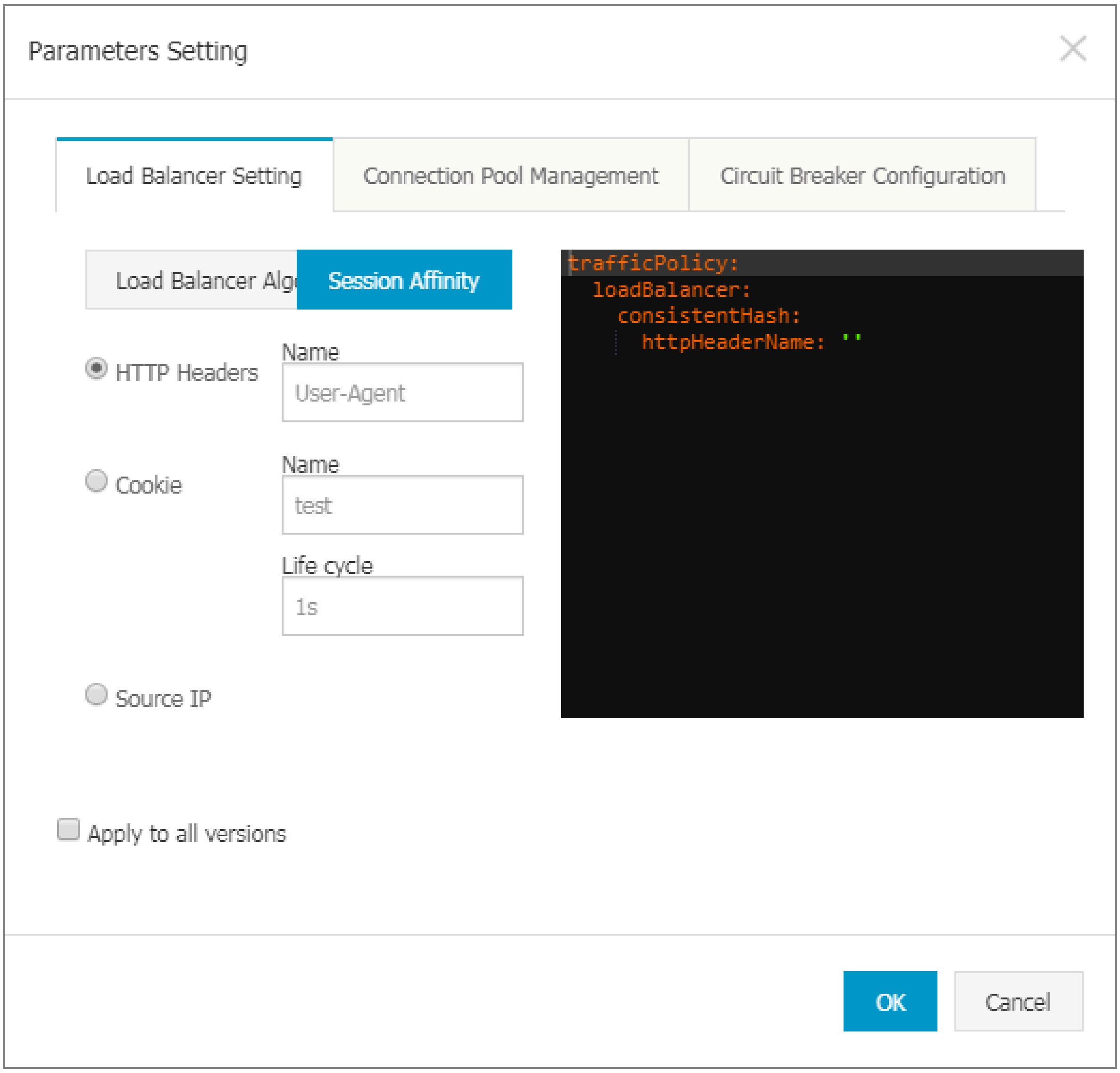
Figure 13-1: Supported load balancing algorithms



Select a type of session affinity according to your needs.

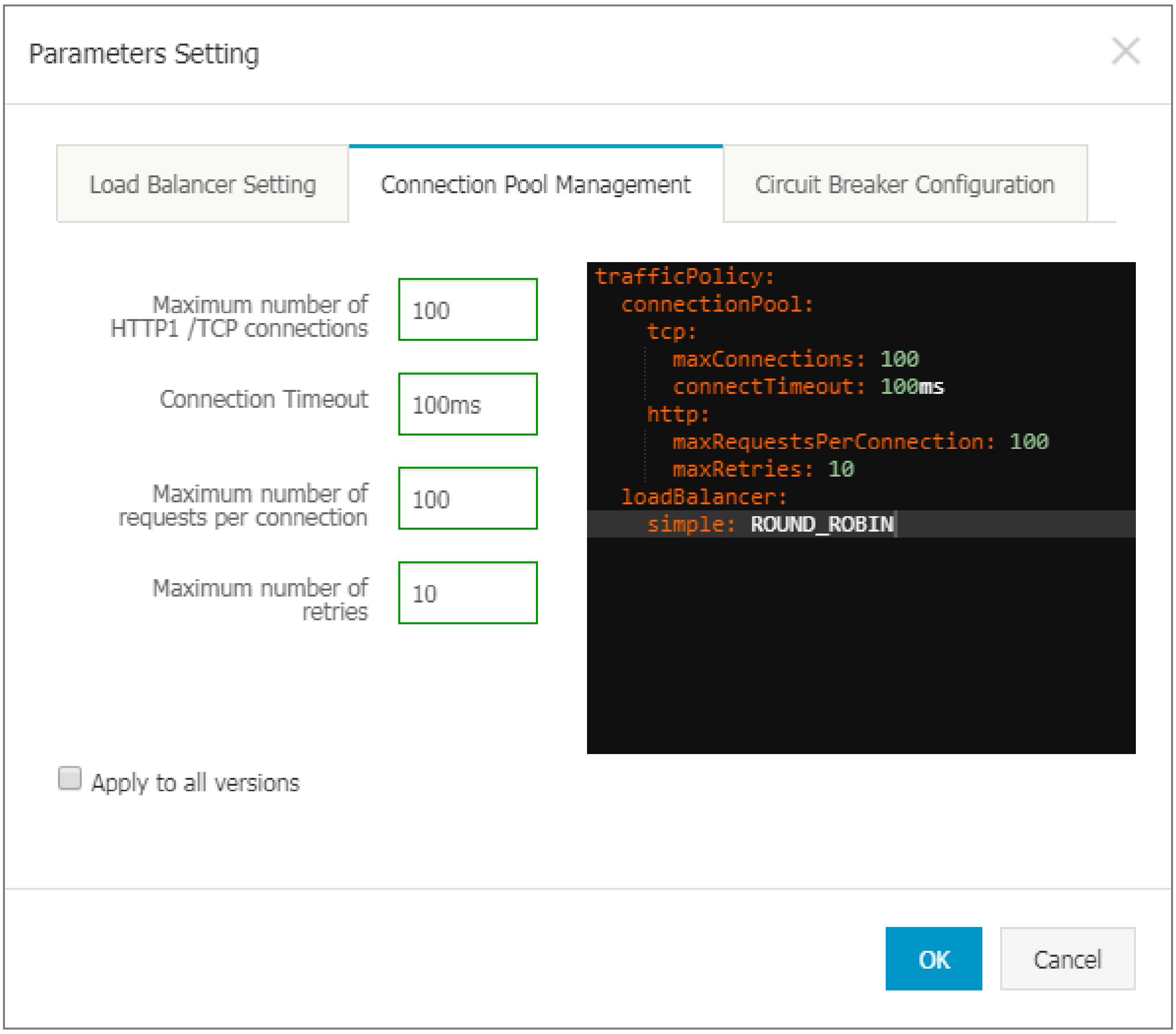
- HTTP Headers: Obtains hashes based on a specific HTTP header.
- Cookie: Obtains hashes based on HTTP cookies.
- Source IP: Obtains hashes based on the source IP addresses.

Figure 13-2: Session affinity options



- **Set a connection pool.**
  - `maxConnections` **: Indicates the maximum number of connections that Envoy will create for all the hosts in the upstream clusters. This parameter applies only to the connections that use the TCP or HTTP/1.1 protocol.**
  - `connectTimeout` **: Indicates the TCP connection timeout. The minimum value of this parameter must be greater than 1 ms. This parameter applies only to the connections that use the TCP or HTTP protocol.**
  - `maxRequestsPerConnection` **: Indicates the maximum number of requests that are destined for a backend through a connection. Setting this parameter to 1 disables the keep-alive feature. This parameter only applies to the connections that use the HTTP/1.1, HTTP/2, or gRPC protocol.**
  - `maxRetries` **: Indicates the maximum number of retries of an HTTP request that is sent to a destination host during a specified period of time. The default**

value of this parameter is 3. This parameter only applies to the connections that use the HTTP/1.1, HTTP/2, or gRPC protocol.

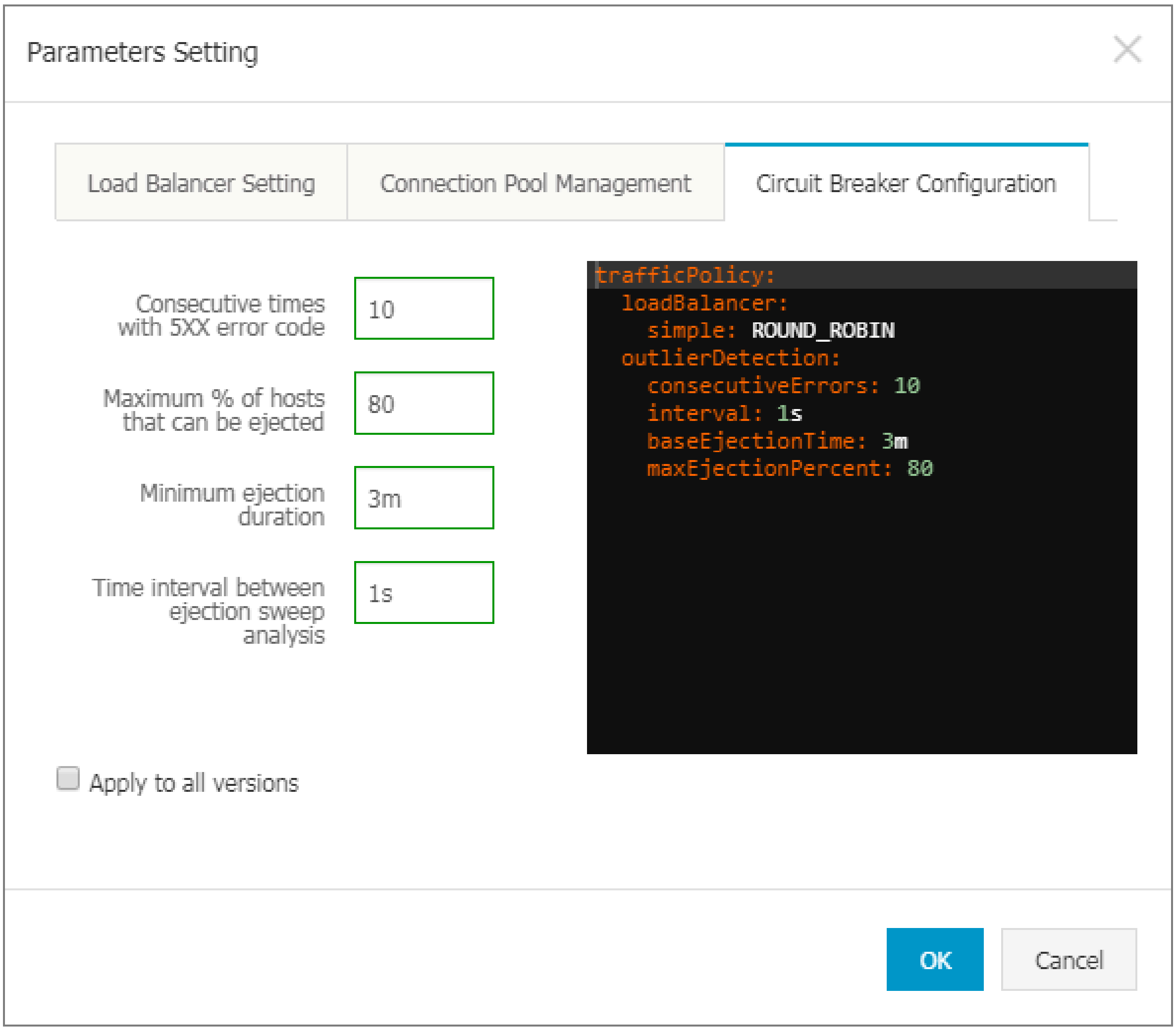


- Set a circuit breaker.
  - `consecutiv eErrors` : Indicates the number of consecutive errors that occur to a host in a specified interval. If the value set is exceeded, the host where the consecutive errors occurred is removed from the connection pool. The default value of this parameter is 5.

> **Note:**
> If the upstream host is accessed through an HTTP connection, a status code of the `5xx` format is identified as an error. If the upstream host is accessed

through a TCP connection, a TCP connection timeout, a connection error, or a connection failure will be identified as an error.

- `maxEjectionPercent`: Indicates the maximum ratio of removable hosts to all the hosts in a load balancing pool of the upstream service. The default value of this parameter is 10%.
- `baseEjectionTime`: Indicates the minimum interval of a time an unhealthy host can be removed from the load balancing pool. The amount of time that an unhealthy host is removed from the load balancing pool is equal to this parameter multiplied by the number of times the host has already been removed. By using this parameter, the amount of time that an unhealthy host is removed from the load balancing pool can be increased automatically.
- `interval`: Indicates the period of time during which the system detects errors. The default value of this parameter is 10s. The minimum value of this parameter is 1 ms.

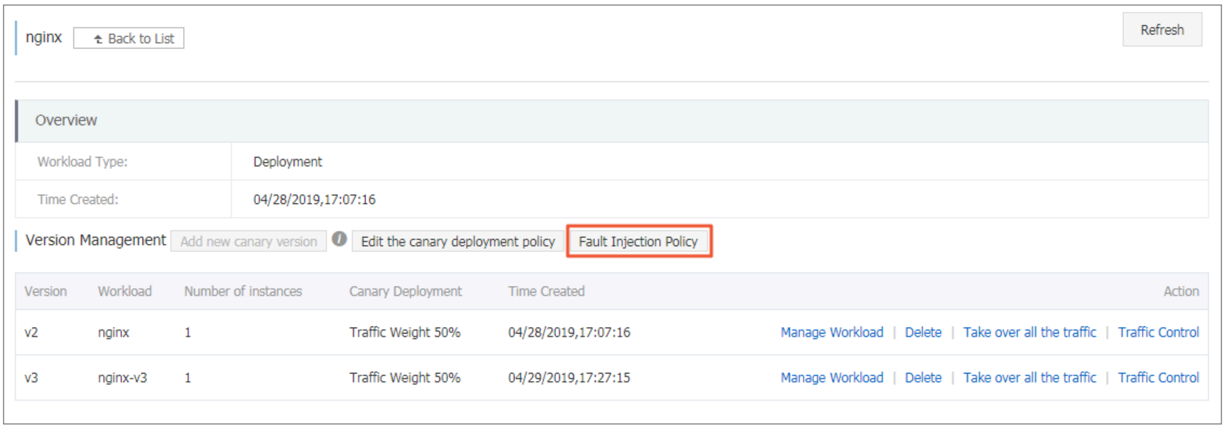


Note:

If you want the preceding settings to take effect for all the versions of the target virtual service, you need to select the Apply to all versions check box.

Inject faults to traffic

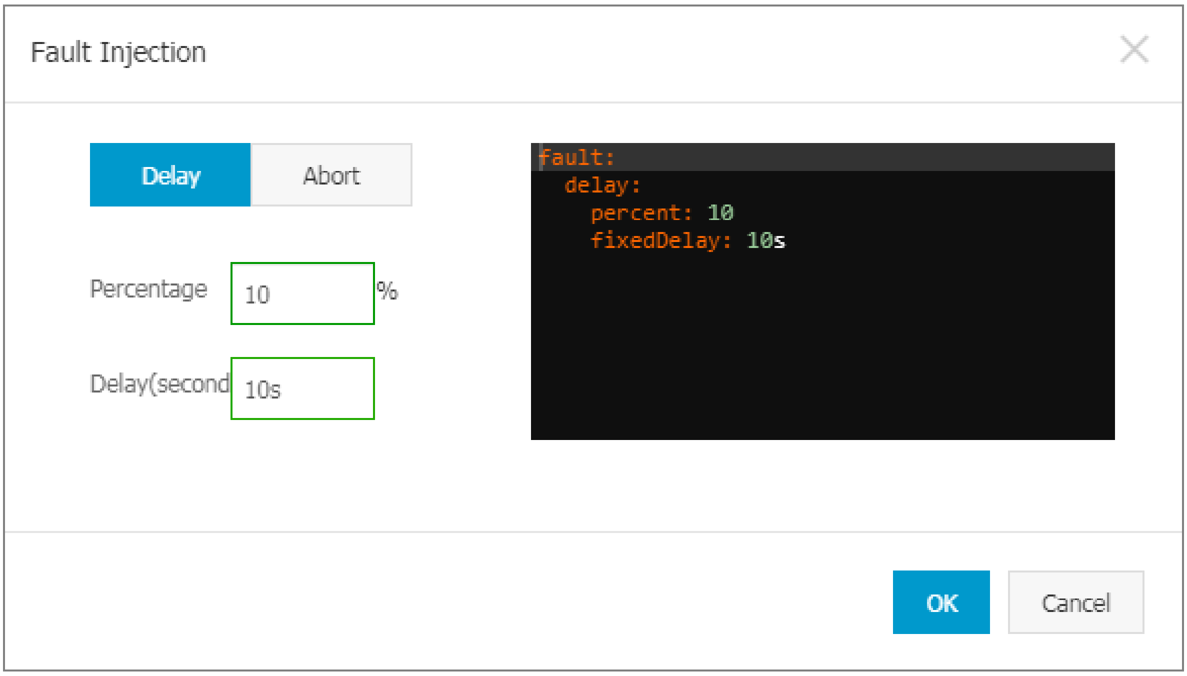
On the page that displays the details of the target virtual service, click Fault Injection Policy.



You can inject two types of faults: delay faults and abort faults. The fault injection feature supports the HTTP protocol.

- Create a fault injection to delay traffic flow.

  This type of fault injection is used to simulate faults, such as network faults and faults caused by upstream service overload.

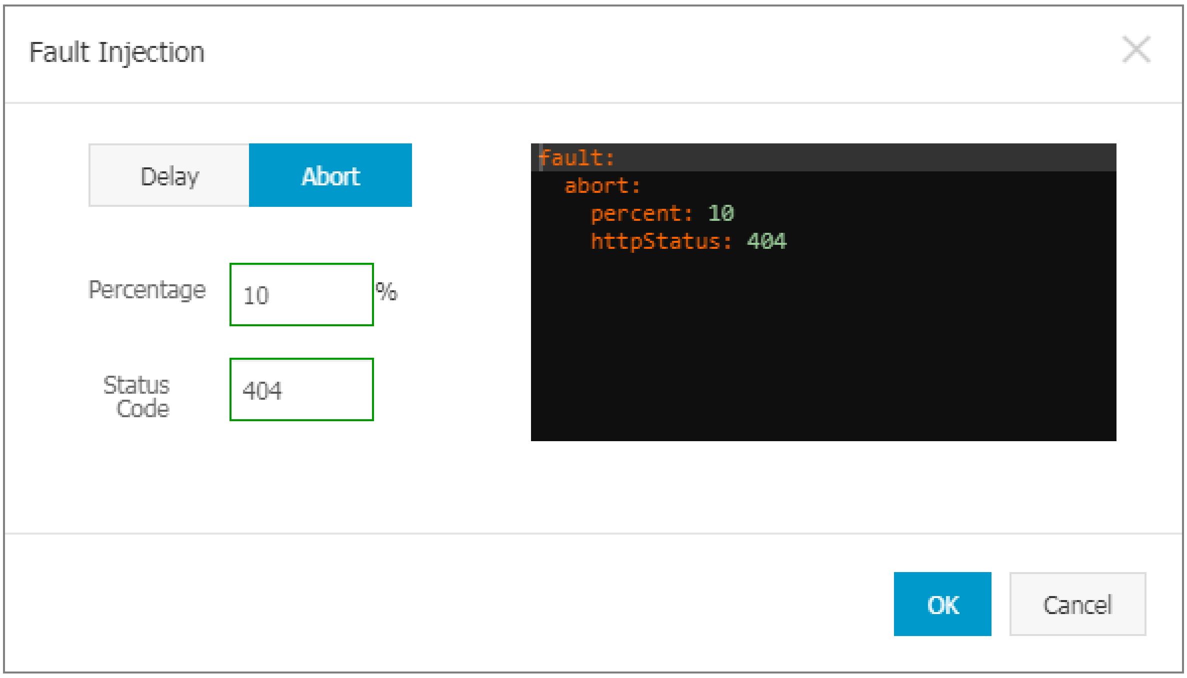


  Create a fault injection rule to delay traffic, set the following parameters.

  - `percent` : Set the ratio of the requests to be delayed as all requests that are forwarded to the requested destination. The value range of this parameter is 0 to 100.
  - `fixedDelay` : Set the delayed time before the specified ratio of requests are forwarded (in seconds by default). You can also set the delayed time in hours, minutes, or milliseconds. This is the required parameter for injecting a delay fault. The minimum value of this parameter is 1ms.

- Create an fault injection that terminates a request that comes from a downstream service and return a corresponding error to the downstream service.

  This type of fault injection is used to simulate a condition where an error code occurs to the upstream service.

  To create an fault injection that terminates a request that comes from a downstream service, set these parameters.

  - `percent` : Set the ratio of the requests that you want to terminate to all the requests that are forwarded to the requested destination. The value range of this parameter is 0 to 100.
  - `httpStatus` : Set the HTTP status code that will be returned to a client service. This is the required parameter for injecting an abort fault.

14 Knative management

14.1 Knative overview

This topic describes the concept of Knative, roles involved in a Knative system, Knative components, and thirty-party add-on supported by Knative.

Background information

Knative is a serverless framework that is based on Kubernetes. The goal of Knative is to provide the cloud-native standard to orchestrate serverless workloads across different platforms. To implement this goal, Knative codifies the best practices around three areas of developing cloud native applications: building container and function, serving and dynamically scaling workloads, and eventing.

Roles involved in the Knative system

- `Developers` : refers to personnel that directly use native Kubernetes APIs to deploy serverless functions, applications, and containers to an auto-scaling runtime.
- `Contributors` : refers to personnel that develop and contribute code and documents to the Knative community.
- `Operators` : refers to personnel that deploy and manage Knative instances by using Kubernetes APIs and tools. Knative can be integrated into any environments that support Kubernetes, such as systems of any enterprises or cloud providers.
- `Users` : refers to personnel that use an Istio gateway to access the target services, or use the eventing system to trigger the serverless service of Knative.

For more information, see Personas involved in Knative.

Components

Knative consists of the following three components:

Build	This component is used to obtain the source code of an application from a code repository, compile the code into container images, and then push the images to an image repository. All these actions occur in the corresponding Kubernetes pods.

Eventing	This component can be used to manage and deliver events. Specifically, Eventing is designed to provide an event model to drive serverless events, including how to connect to an external event source, register and subscribe events, and filter events. The event model decouples event producers and event consumers. This means that any producer can generate events before a consumer starts to listen to the events, and any event consumer can listen events before producers start to generate the events.
Serving	This component can be used to manage serverless workloads. It provides the request-driven capability to auto scale workloads. If no request is needed to be processed, Serving can help to scale workload instances to zero instance. For advanced scenarios, workload instances can be scaled to any required number without limitation. In addition, this component supports the function of granary deployment.

Third-party add-on

Knative supports the GitHub add-on for using the GitHub event source.

14.2 Deploy Knative on a Kubernetes cluster

This topic descries how to deploy Knative on a Kubernetes cluster that is created with Alibaba Cloud Container Service for Kubernetes (ACK).

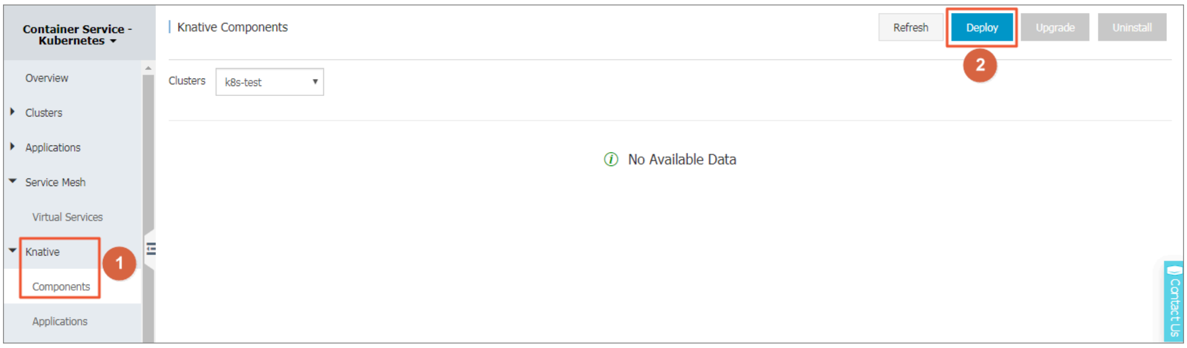
Prerequisites

- A Kubernetes cluster supported by at least three Worker nodes is created with ACK. For more information, see Create a Kubernetes cluster.
- Istio is deployed on the Kubernetes cluster. For more information, see Deploy Istio on a Kubernetes cluster.
- The Kubernetes cluster must be a standard dedicated or managed cluster that runs on Kubernetes V1.10 and later.

Procedure

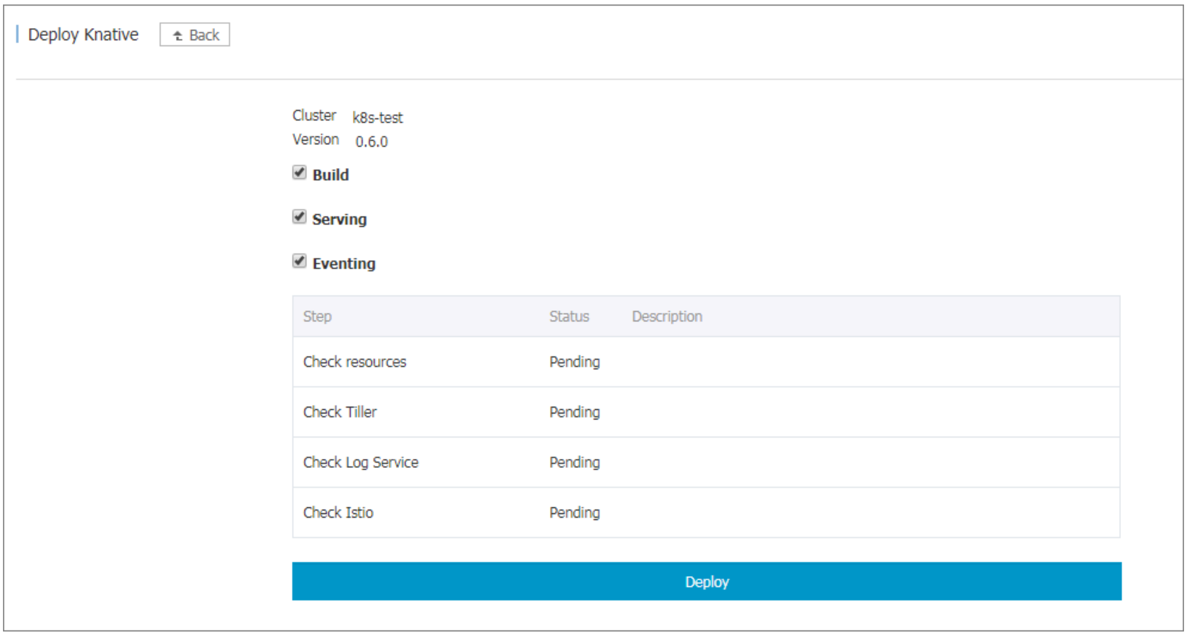
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Knative > Components.

3. In the upper-right corner, click Deploy.



4. Select the required components for Knative as needed, and then click Deploy.

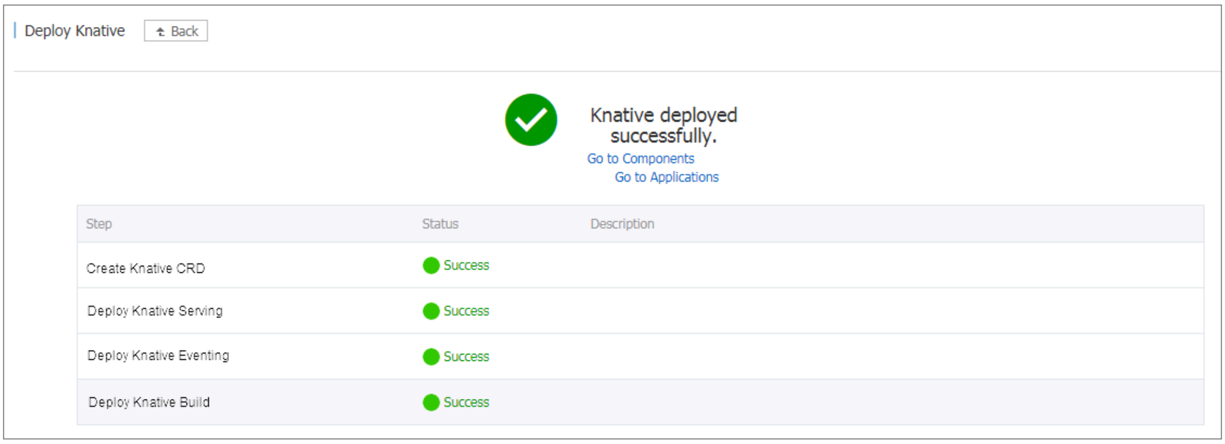
- Bulid : This component is used to obtain the source code of an application from a code repository, compile the code into container images, and then push the images to an image repository.
- Serving: This component can be used to manage serverless workloads. It can work well with the Eventing component. It provides the request-driven capability to auto scale workloads. If no request is needed to be processed, Serving can help to scale workload instances to zero instance.
- Eventing : This component can be used to manage events.



Verify the result

On the Deploy Knative page, verify that Knative is deployed.

- To view component information, click Go to components.
- To view operations related to applications, click Go to applications.



14.3 Deploy a Knative component

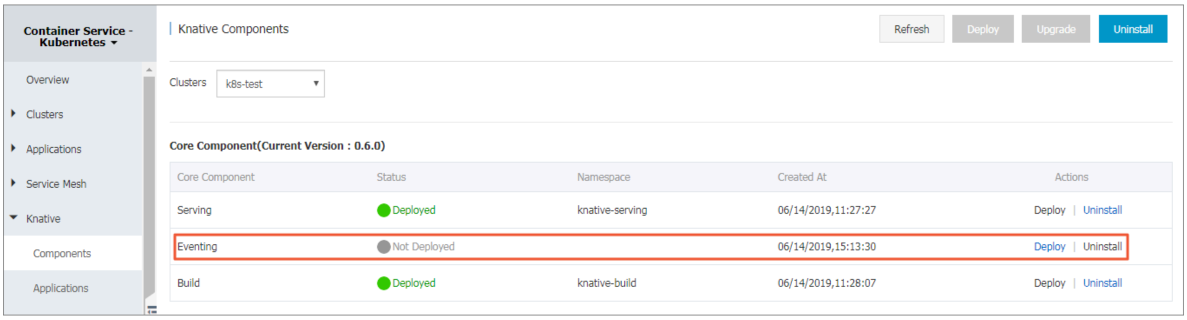
This topic describes how to deploy a Knative component by deploying the component Eventing as an example. If you did not select a target component when you deployed Knative, you can follow the procedure described in this topic.

Prerequisites

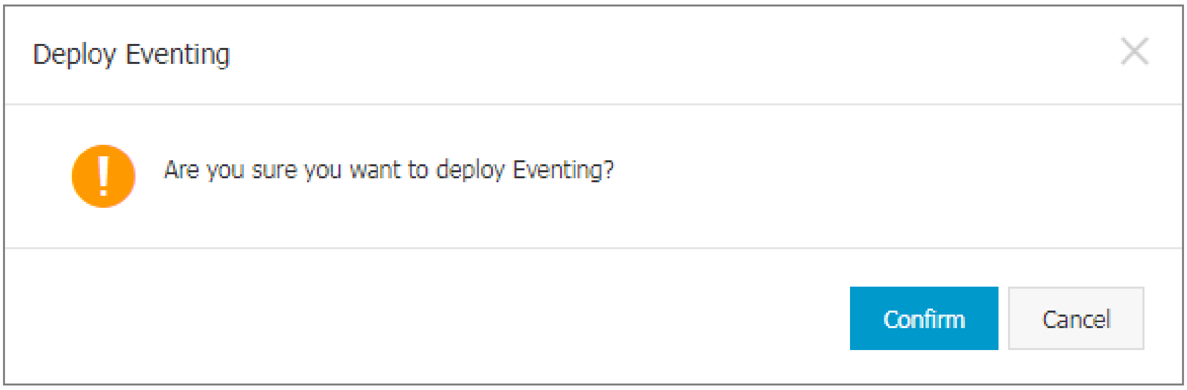
- A Kubernetes cluster is created with ACK. For more information, see Create a Kubernetes cluster.
- Knative is deployed on the Kubernetes cluster. For more information, see Deploy Knative on a Kubernetes cluster.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Knative > Components.
3. Find the target component. Then, in the Actions column, click Deploy.

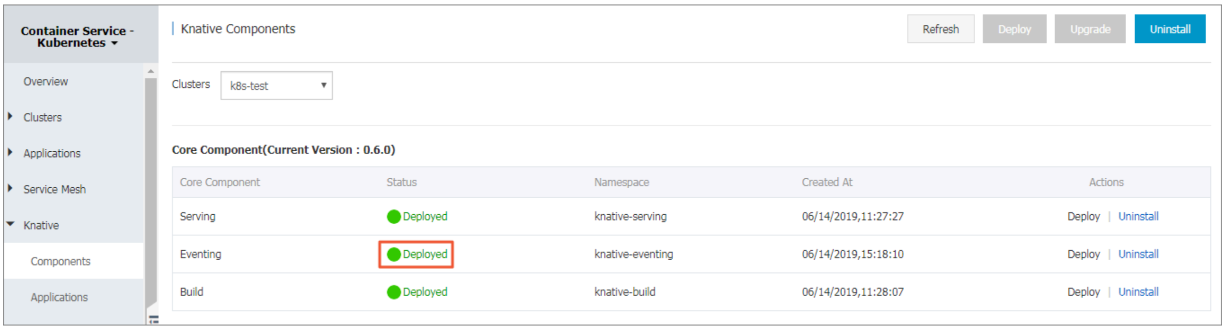


4. In the displayed dialog box, click Confirm.



Verify the result

On the Knative Components page, verify that the status of the target component is `Deployed`.



14.4 Use Knative to deploy a Hello World application

This topic describes how to use Knative to deploy an application.

Prerequisites

- A Kubernetes cluster is created with ACK. For more information, see Create a Kubernetes cluster.
- Knative is deployed on the Kubernetes cluster. For more information, see Deploy Knative on a Kubernetes cluster.
- The Serving component is deployed on the Kubernetes cluster. For more information, see Deploy a Knative component.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Knative > Services.

3. In the upper-righter corner, click Create Service.
4. Set the following parameters:
   - Service Name: Set the service name. In this example, the service name is set as `helloworld - go` .
   - Image Name: Select an image by clicking Select Image, or in the box, enter a private registry that must be in the format of *`domainname / namespace / imagename : tag`* .

     Note:
     In this example, the registry *`registry . cn - hangzhou . aliyuncs . com / knative - sample / helloworld - go`* is used.
   - Image Version: Select an image by clicking Select Image Version, or in the box, enter an image version. By default, if you do not set this parameter, the latest image version is used. In this example, the image version is set as `73fbdd56` .
   - Environment Variables: Set an environment variable by entering a key-value pair. In this example, the environment variable is set as `TARGET = Knative` .
5. Click Create.

   On the Knative Service Manage page, the created service is then displayed.

Verify the result

To access the created Knative service through its URL, follow these steps:

1. In the left-side navigation pane under Container Service-Kubernetes, choose Knative > Component to view the IP address of the gateway of the Knative service.

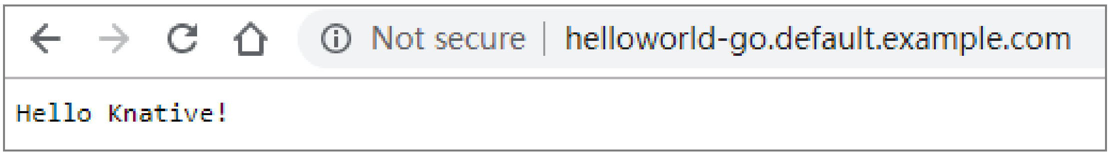
2. In the host file of your local host, associate the IP address of the gateway and the domain name of the Knative service with your local host by adding the following information:

```
the IP address of the gateway + the domain name
```

The following is a sample:

```
47 . 95 . XX . XX   helloworld - go . default . example . com
```

3. In your browser, enter the URL `http :// helloworld - go . default . example . com` to access the Knative service.



14.5 Uninstall a Knative component

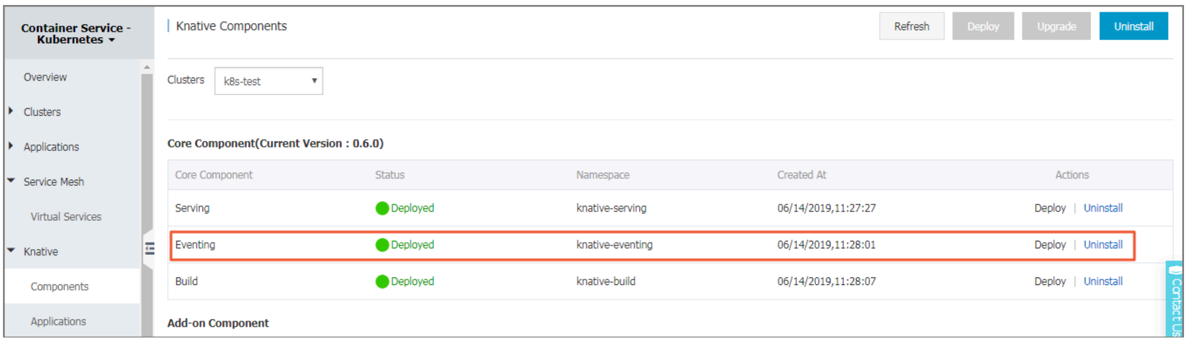
This topic describes how to uninstall a Knative component. In this topic, an example component Eventing is uninstalled.

Prerequisites

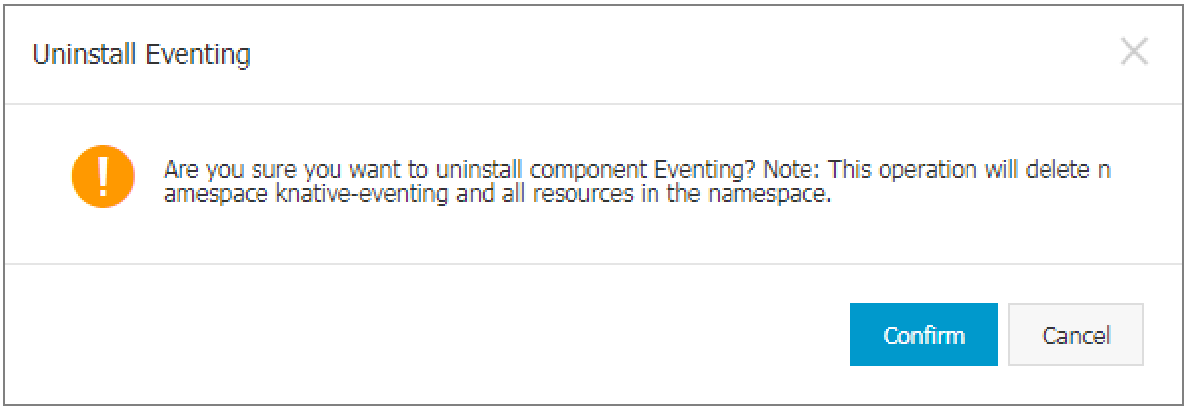
- A Kubernetes cluster is created with ACK. For more information, see Create a Kubernetes cluster.
- Knative is deployed on the Kubernetes cluster. For more information, see Deploy Knative on a Kubernetes cluster.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Knative > Components.
3. Find the target component. Then, in the Actions column, click Uninstall.

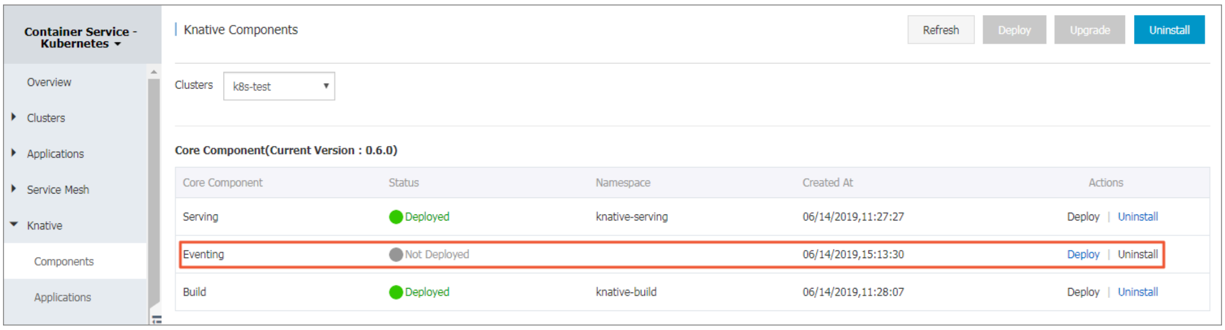


4. In the displayed dialog box, click Confirm.



Verify the result

On the Knative Components page, verify that the status of the target component is `Not deployed`.



14.6 Uninstall Knative

This topic descries how to uninstall Knative from a Kubernetes cluster that is created with Alibaba Cloud Container Service for Kubernetes (ACK).

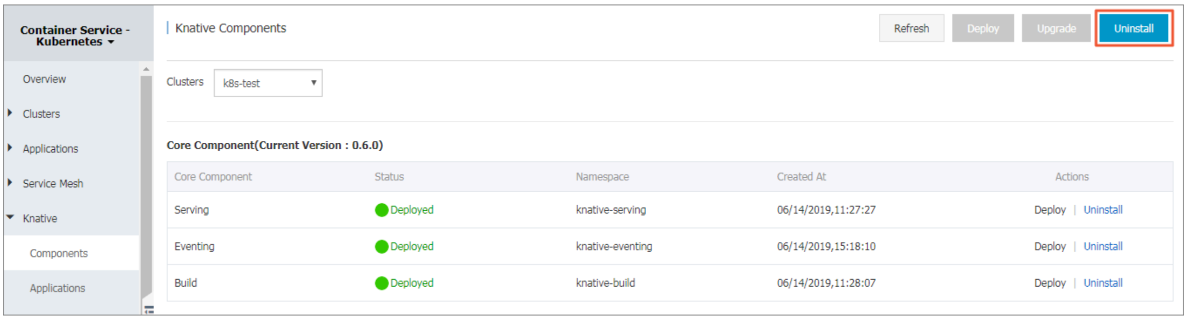
Prerequisites

Knative is deployed on a Kubernetes cluster that is created with ACK. For more information, see Deploy Knative on a Kubernetes cluster.

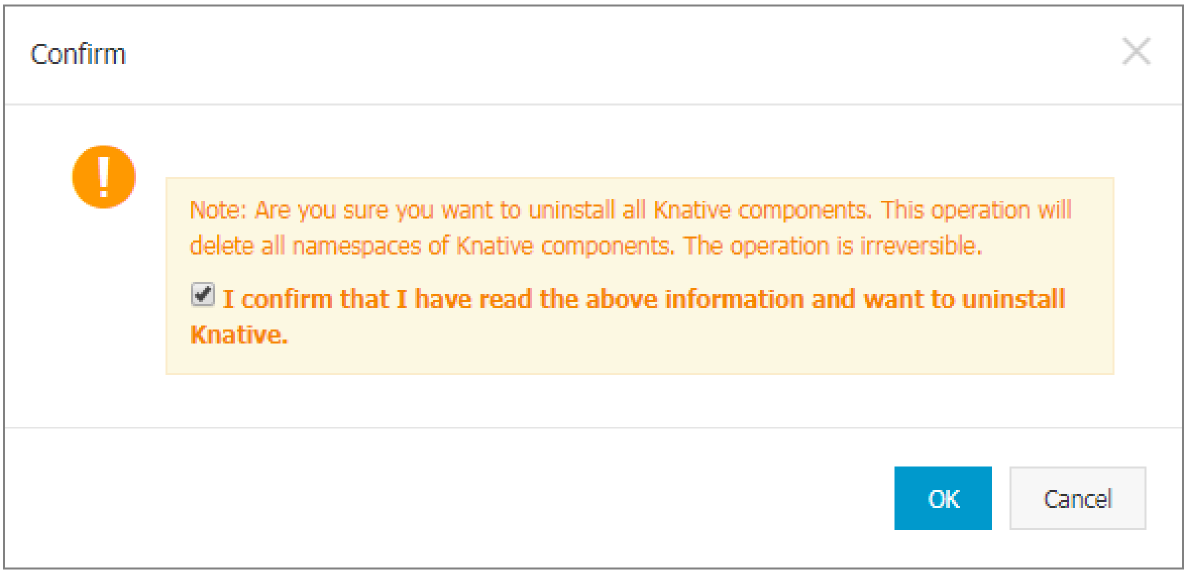
Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Knative > Components.

3. In the upper-right corner, click Uninstall.

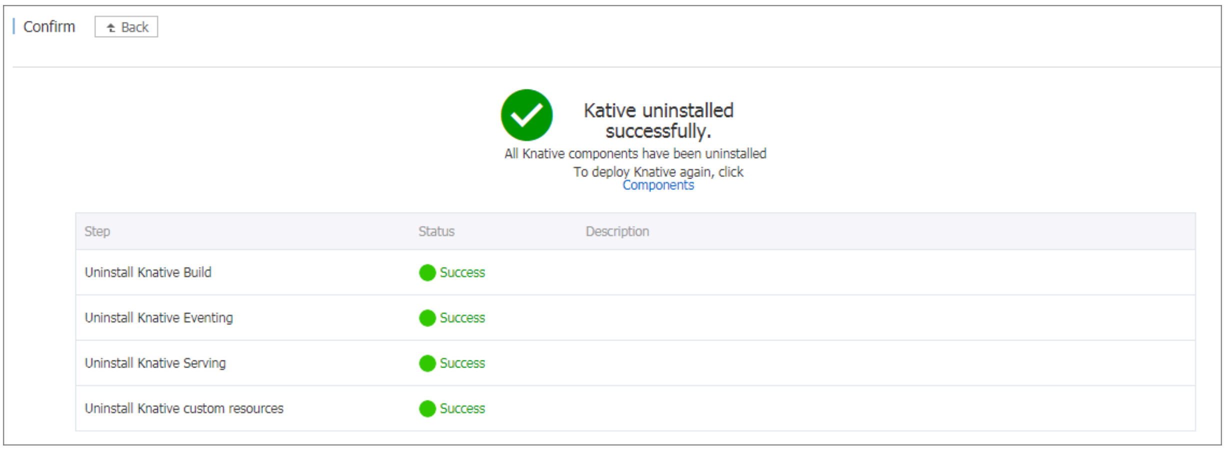


4. In the displayed dialog box, select I confirm that I have read the above information and want to uninstall Knative, and then click OK.



Verify the result

On the displayed Confirm page, verify that Knative is uninstalled.



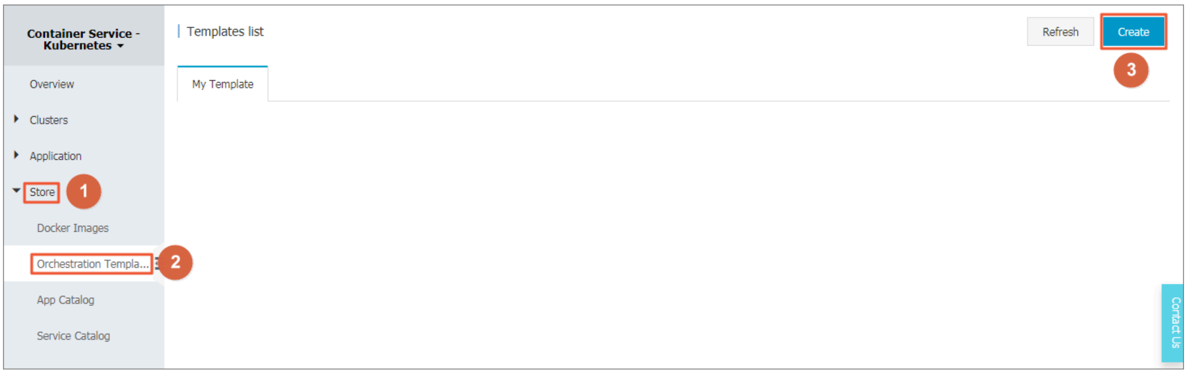
15 Template management

15.1 Create an orchestration template

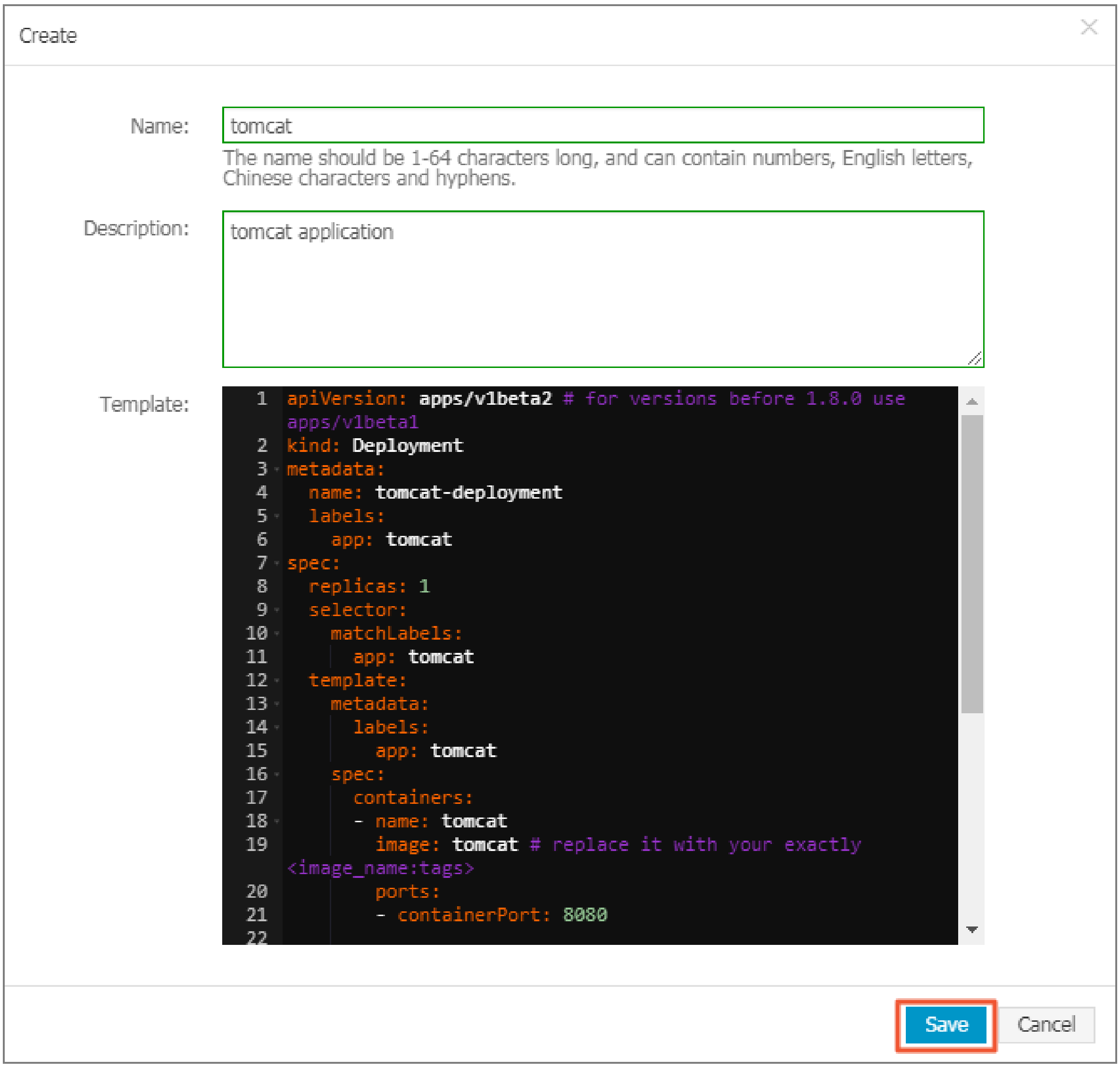
You can use multiple methods to create orchestration templates through the Container Service console.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Store > Orchestration Templates. Then, in the upper-right corner, click Create.



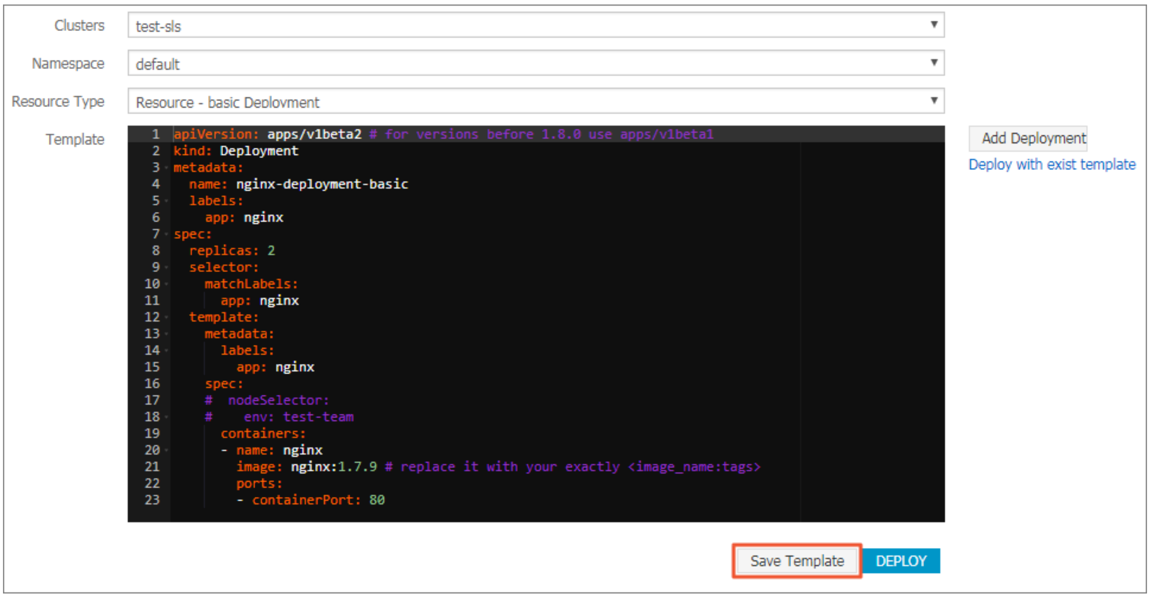
3. In the displayed dialog box, configure the orchestration template, and then click Save. In this example, build a tomcat application template that contains a deployment and a service.
   - Name: Set the template name.
   - Description: Enter the description for the template. This parameter is optional.
   - Template: Configure the template that conforms to Kubernetes yaml syntax rules. The template can contain multiple resource objects that are separated by `---`.



4. After the template is created, the Template List page is displayed. You can see the template under My Template.

5. Optional: In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments. Then, in the upper-right corner, click Create by Template. Save one of orchestration templates built-in Container Service as your custom template.

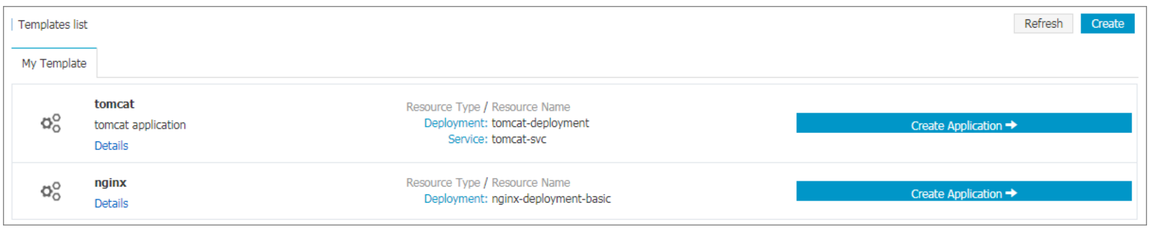
   a) Select a built-in template and click Save Template.



   b) In the displayed dialog box, configure the name, description, and template. After completing the configurations, click Save.

   > **Note:**
   > You can modify the built-in template.

   c) Choose Store > Orchestration Templates, the created template is displayed under My Template.



What's next

You can quickly create an application by using the orchestration template under My Template.

15.2 Edit an orchestration template

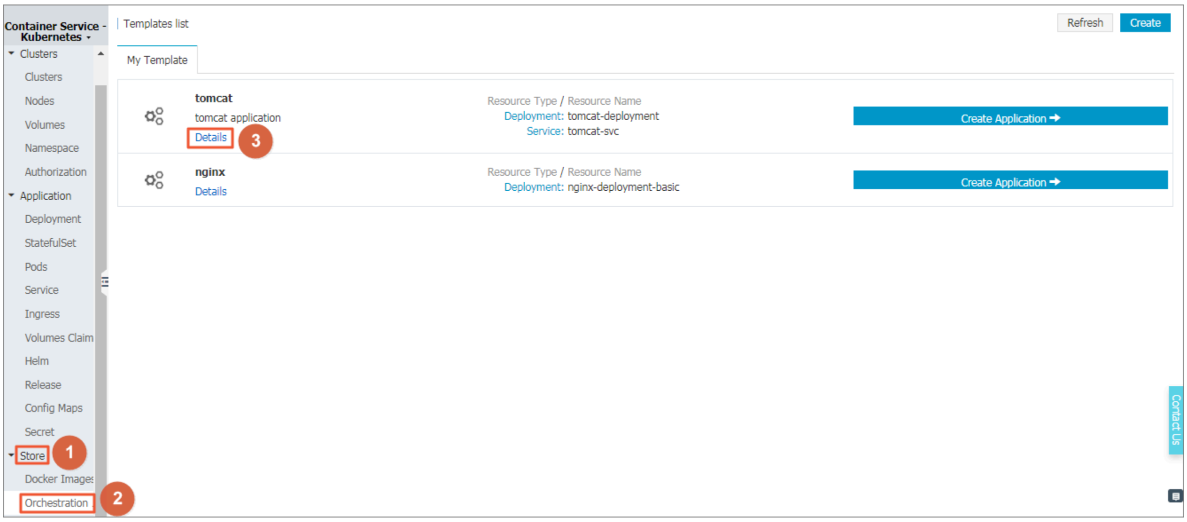
You can edit an orchestration arrangement template.

Prerequisites

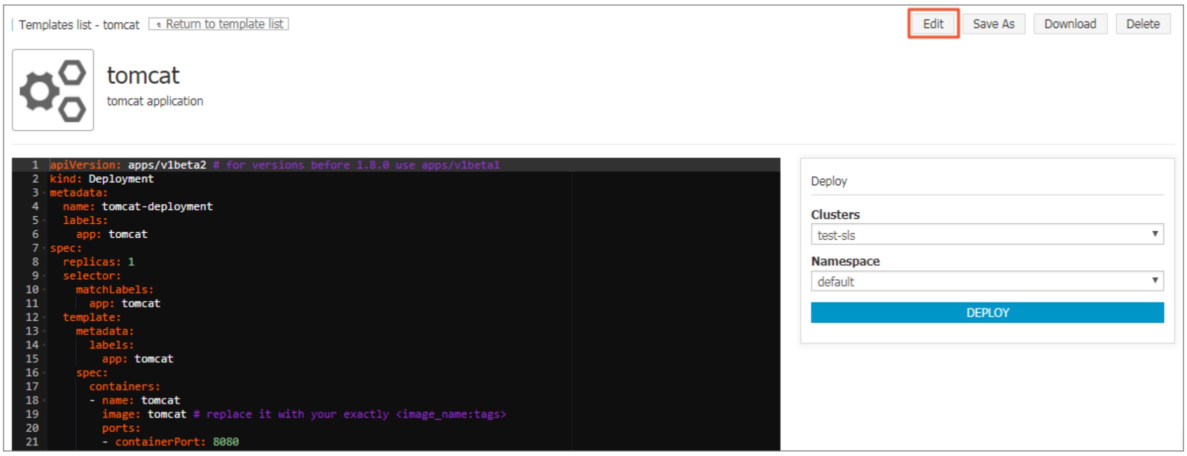
You have created an orchestration template, see Create an orchestration template.

Procedure

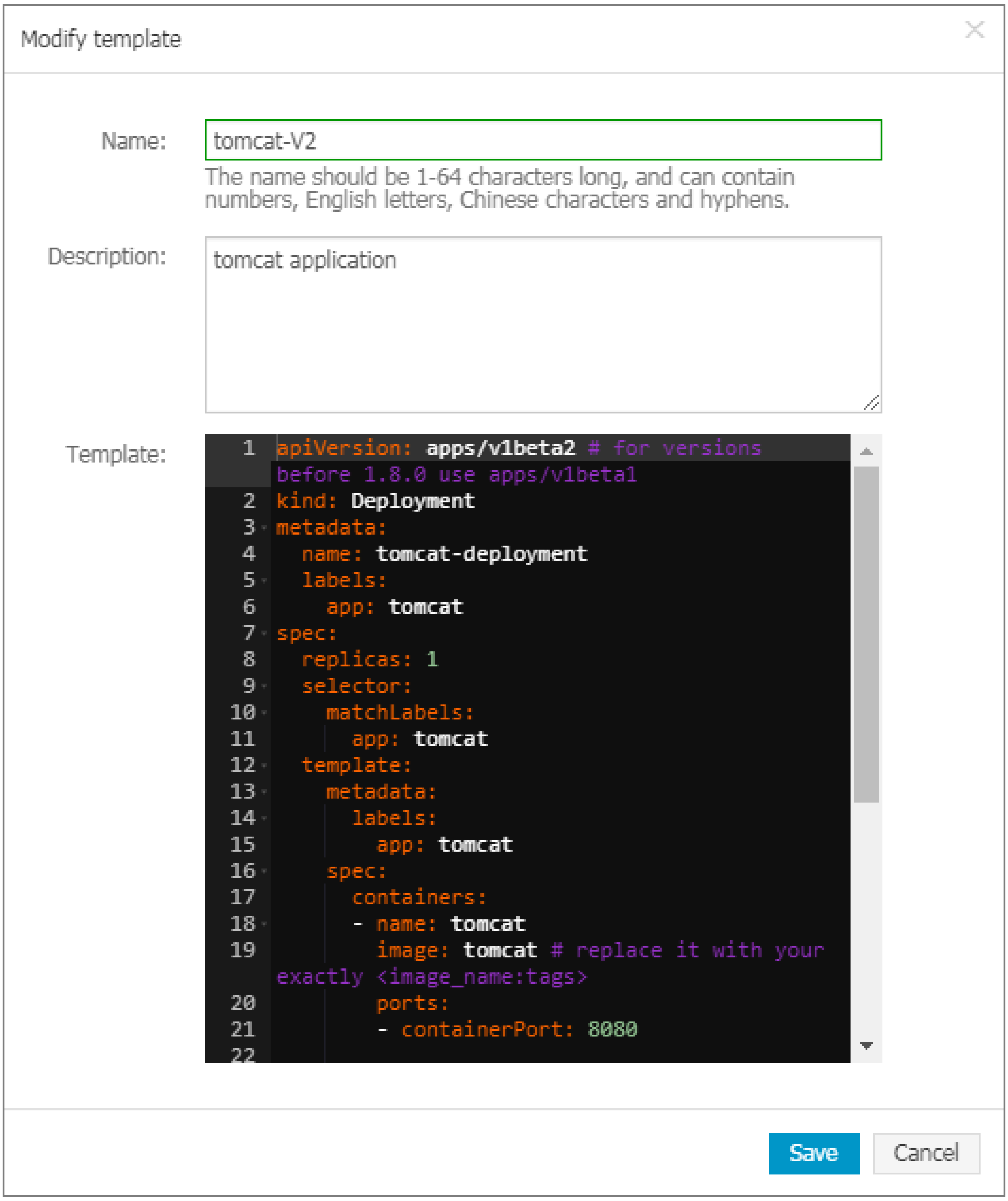
1. Log on to the Container Service console.
2. Under Kubernetes, click Store > Orchestration Templates. Existing orchestration templates are displayed under My Template.
3. Select a template and click Details.



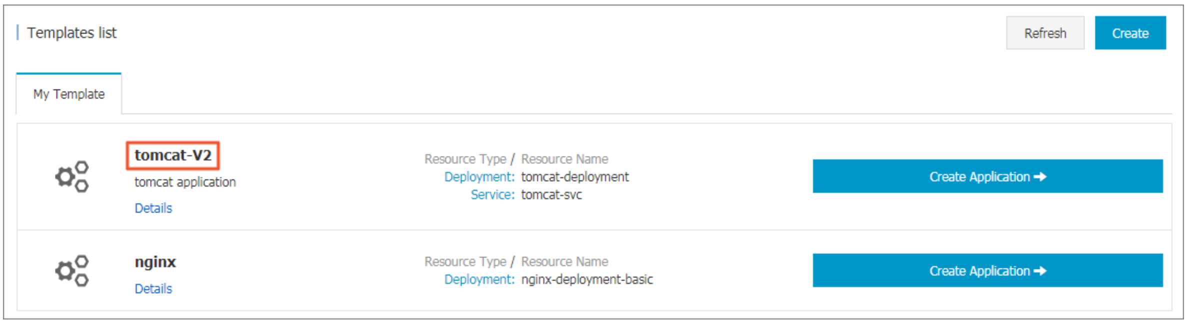
4. Click Edit in the upper-right corner.



5. In the displayed dialog box, edit the name, description, and template, and click Save.



6. Back to the Template List page, under My Template, you can see the template is changed.



15.3 Save an existing orchestration template as a new one

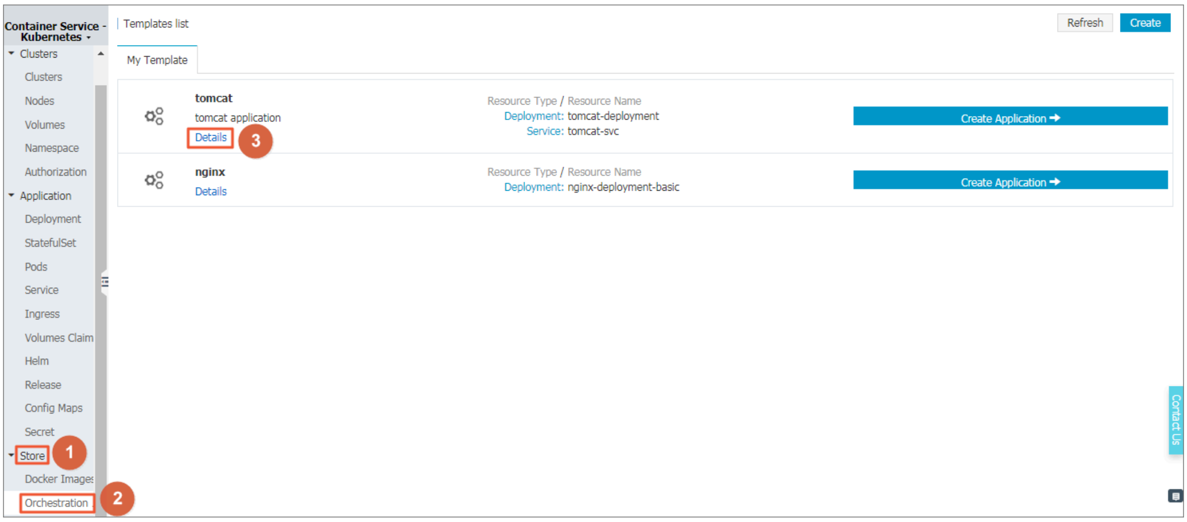
You can save an existing template as a new one.

Prerequisites

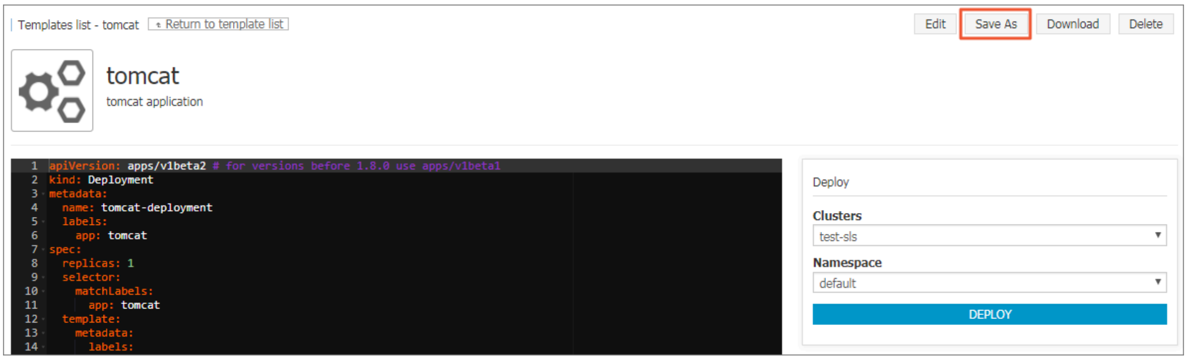
You have created an orchestration template, see Create an orchestration template.

Procedure

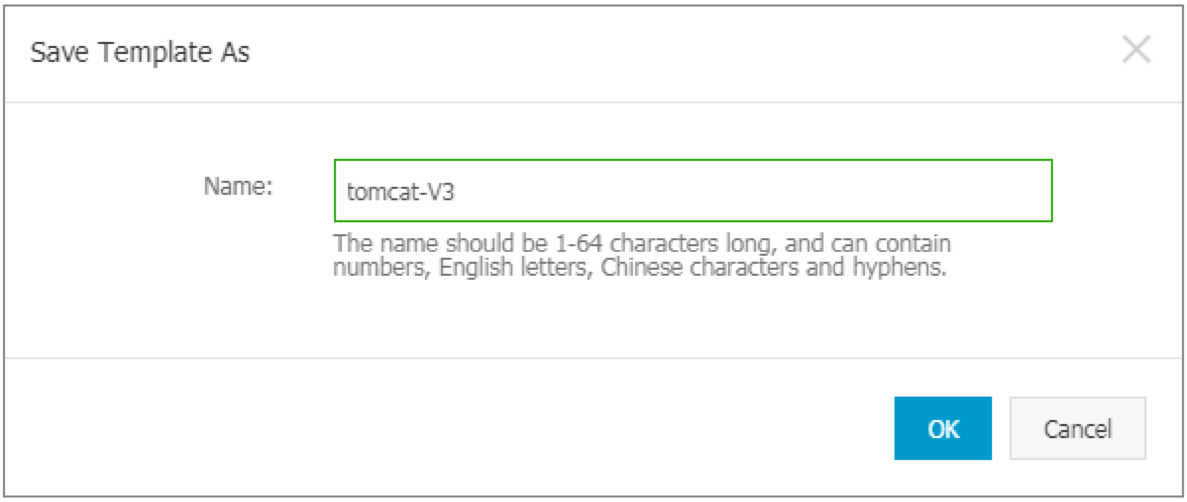
1. Log on to the Container Service console.
2. Under Kubernetes, click Store > Orchestration Templates. Existing orchestration templates are displayed under My Template.
3. Select a template and click Details.



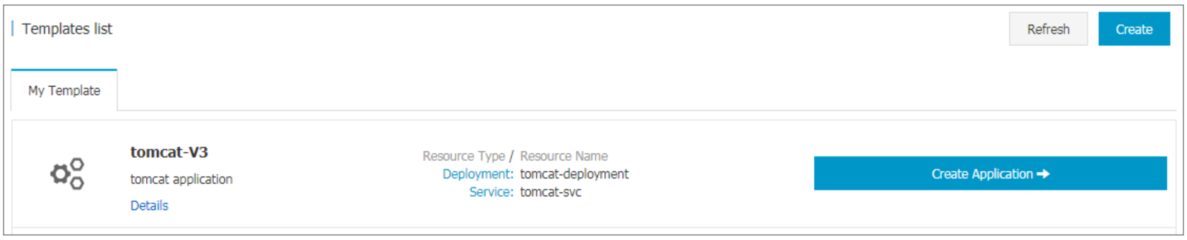
4. You can modify the template and click Save as in the upper-right corner.



5. In the displayed dialog box, configure the template name and click OK.



6. Back to the Template List page, you can see that the saved template is displayed under My Template.



15.4 Download an orchestration template

You can download an existing orchestration template.

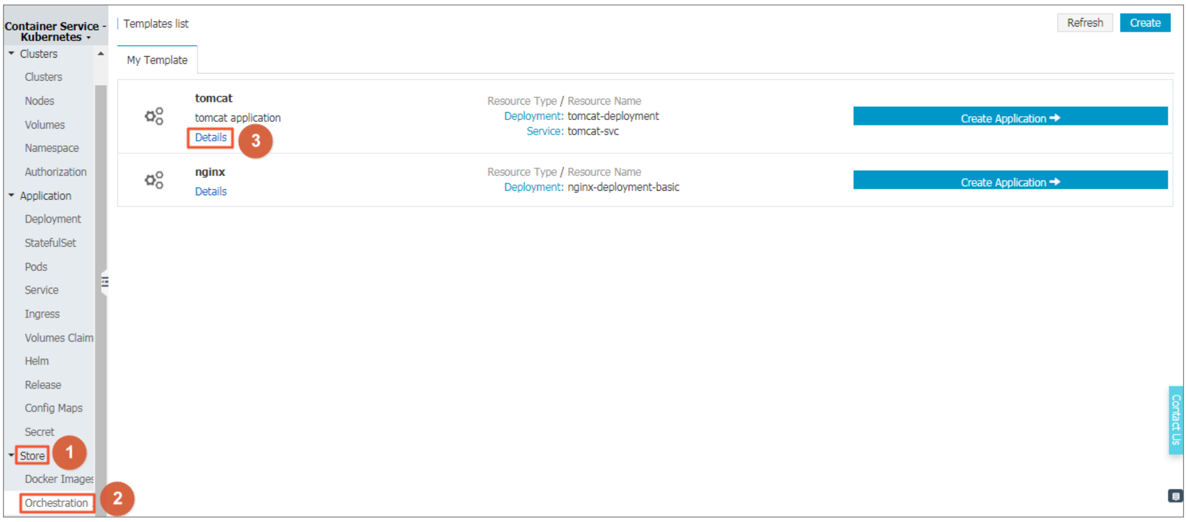
Prerequisites

You have created an orchestration template, see Create an orchestration template.

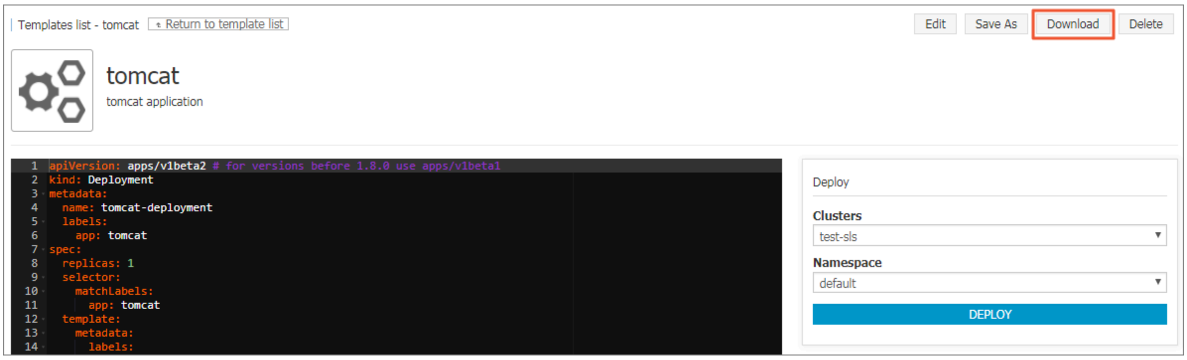
Procedure

1. Log on to the Container Service console.

2. Under Kubernetes, click Store > Orchestration Templates. Existing orchestration templates are displayed under My Template.
3. Select a template and click Details.



4. Click Download in the upper-right corner, a template file with yml suffix is downloaded immediately.



15.5 Delete an orchestration template

You can delete an orchestration template that is no longer needed.

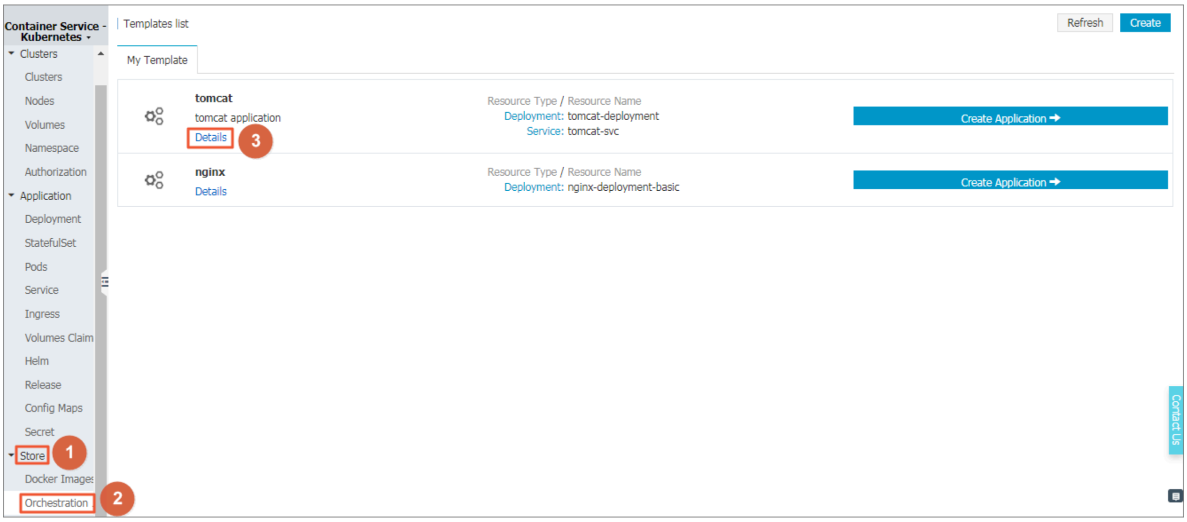
Prerequisites

You have created an orchestration template, see Create an orchestration template.

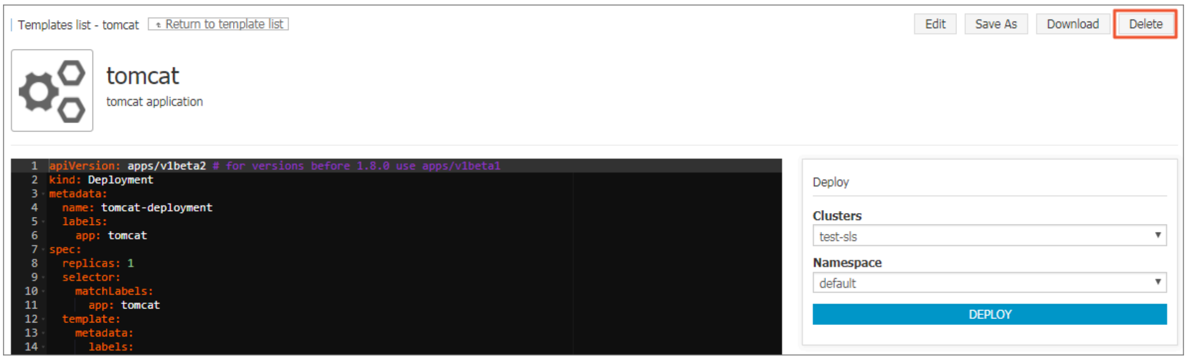
Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Store > Orchestration Template. Existing orchestration templates are displayed under My Template on the Template list page.

3. Select a template and click Detail.



4. On the detail page of the template, you can click Delete in the upper-right corner.



5. Click Confirm in the displayed dialog box.

16 App catalog management

16.1 App catalog overview

Microservice is the theme of container era. The application microservice brings great challenge to the deployment and management. By dividing a large single applicatio n into several microservices, the microservice can be independently deployed and extended so as to realize the agile development and fast iteration. Microservice brings great benefits to us. However, developers have to face the management issues of the microservices, such as the resource management, version management, and configuration management. The number of microservices is large because an application is divided into many components that correspond to many microservices.

For the microservice management issues under Kubernetes orchestration, Alibaba Cloud Container Service introduces and integrates with the Helm open-source project to help simplify the deployment and management of Kubernetes applications.

Helm is an open-source subproject in the Kubernetes service orchestration field and a package management tool for Kubernetes applications. Helm supports managing and controlling the published versions in the form of packaging softwares, which simplifies the complexity of deploying and managing Kubernetes applications.

Alibaba Cloud app catalog feature

Alibaba Cloud Container Service app catalog feature integrates with Helm, provides the Helm-related features, and extends the features, such as providing graphic interface and Alibaba Cloud official repository.

The chart list on the App Catalog page includes the following information:

- Chart name: A Helm package corresponding to an application, which contains the image, dependencies, and resource definition required to run an application.
- Version: The version of the chart.
- Repository: The repository used to publish and store charts, such as the official repository stable and incubator.

The information displayed on the details page of each chart may be different and include the following items:

- Chart introduction
- Chart details
- Prerequisites for installing chart to the cluster, such as pre-configuring the persistent storage volumes (pv)
- Chart installation commands
- Chart uninstallation commands
- Chart parameter configurations

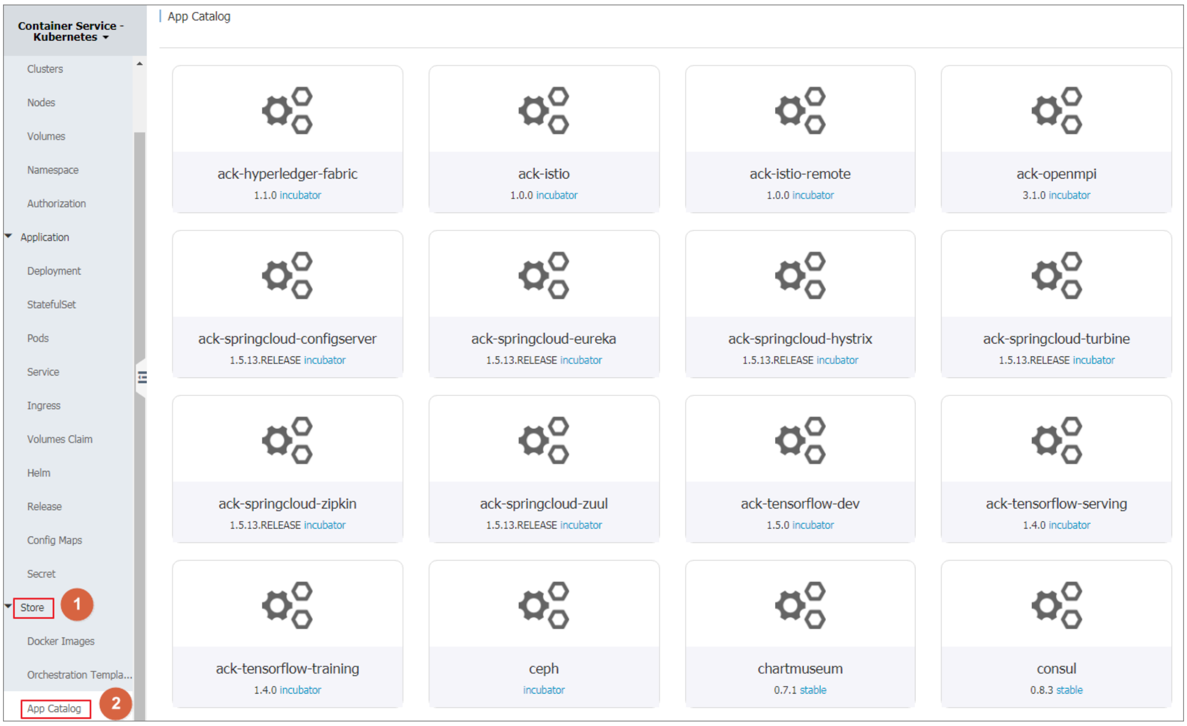
Currently, you can deploy and manage the charts in the app catalog by using the Helm tool. For more information, see Simplify Kubernetes application deployment by using Helm.

16.2 View app catalog list

Procedure

1. Log on to the Container Service console.
2. Under Kubernetes, click Store > App Catalog in the left-side navigation pane.

   View the charts on the App Catalog page, each of which corresponds to an application, containing some basic information such as the application name, version, and source repository.



What's next

You can click to enter a chart and get to know the detailed chart information. Deploy the application according to the corresponding information by using the Helm tool. For more information, see Simplify Kubernetes application deployment by using Helm.

17 Service catalog management

17.1 Overview

Applications running on the cloud platform need some basic services such as databases, application servers, and other generic basic softwares. For example, a WordPress application, as a Web application, needs a database service (such as MariaDB) in the backend. Traditionally, you can create the MariaDB service on which the application depends in the WordPress application orchestration, and integrate the MariaDB service with the Web application. To develop applications on the cloud in this way, developers must spend time and energy deploying and configuring the dependent infrastructure softwares, which increases the costs of hosting and migrating applications.

Alibaba Cloud Container Service supports and integrates with the service catalog function. The service catalog function aims to access and manage the service brokers, which allows applications running in Kubernetes clusters to use the managed services offered by service brokers. A series of infrastructure softwares are supported by the service catalog function, which allows the developers to use these softwares as services and focus on the applications, the core of the development, without concerning about the availability and scalability of the softwares or managing the softwares.

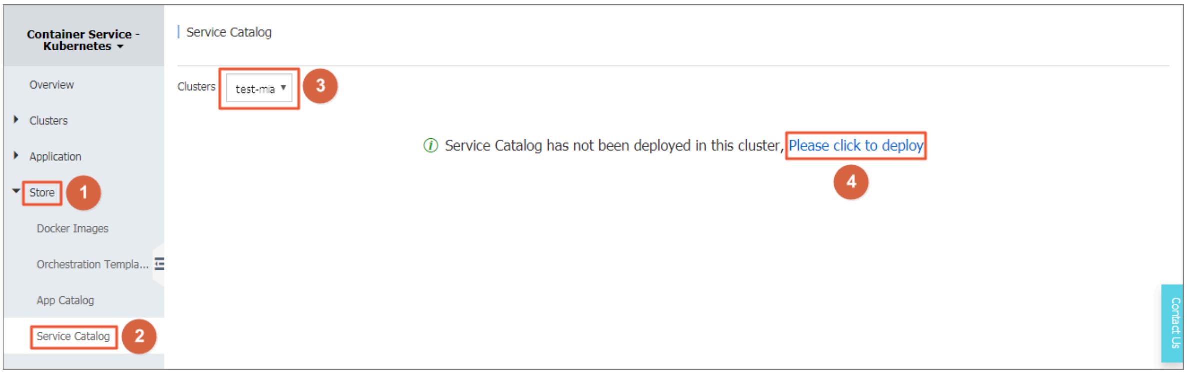
The service catalog uses the Open service broker API of Kubernetes to communicate with service brokers, acting as an intermediary for the Kubernetes API server to negotiate the initial provisioning and obtain the credentials necessary for the applications to use the managed services. For more information about the implementation principle of the service catalog, see Service catalog.

17.2 Enable service catalog function

Procedure

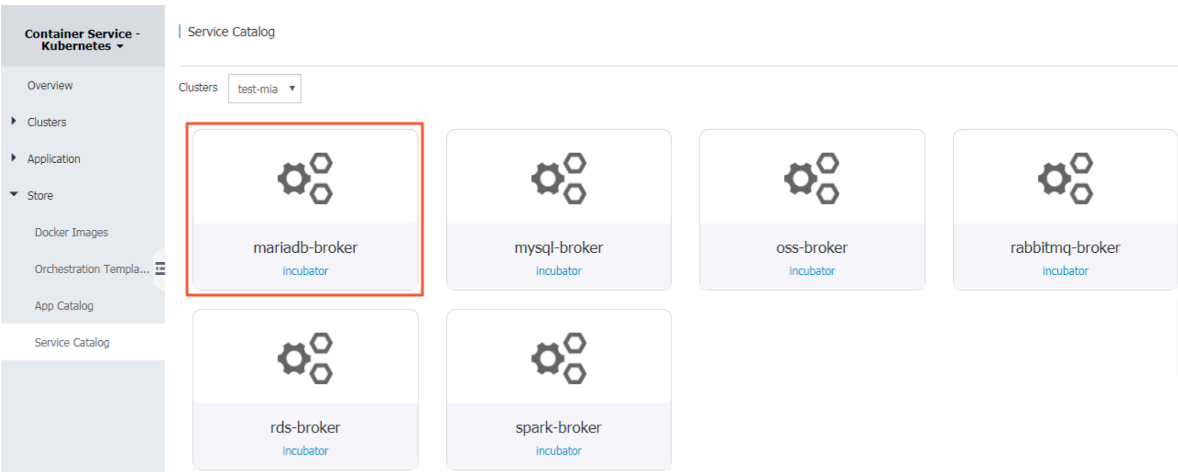
1. Log on to the Container Service console.
2. Under Kubernetes, click Store > Service Catalog in the left-side navigation pane. Select the cluster from the Cluster drop-down list in the upper-right corner.

3. If you have not deployed the service catalog, click to install the service catalog as instructed on the page.

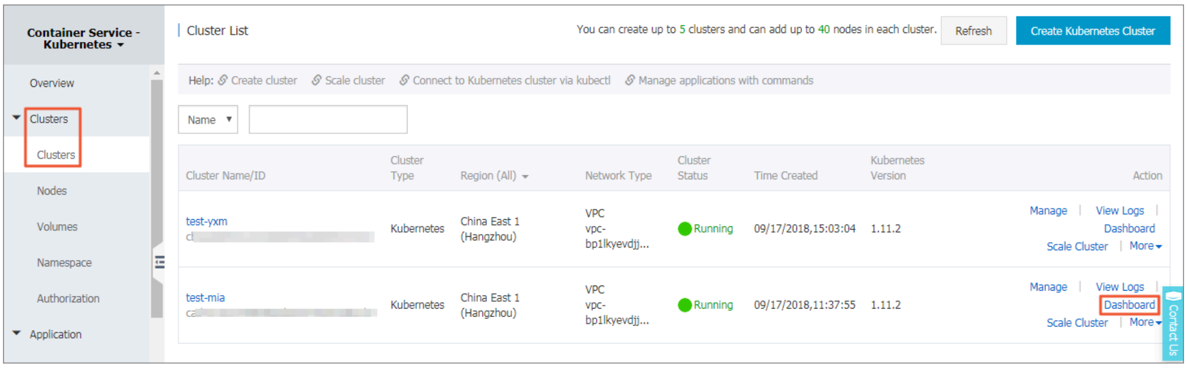


4. After the installation, the service broker, which is installed by default, is displayed on the Service Catalog page. You can click the mariadb-broker to view the details.

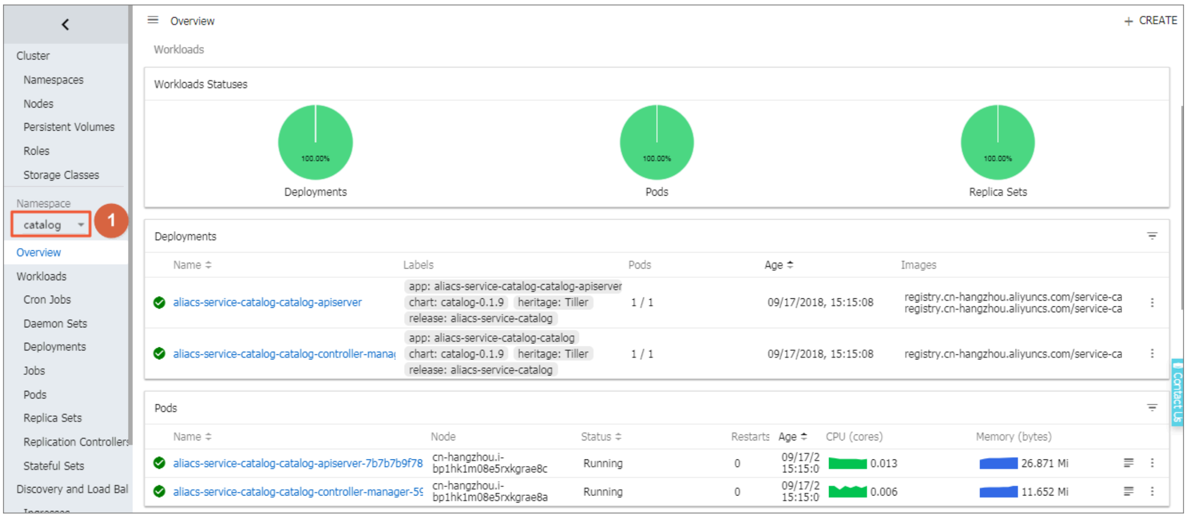
> **Note:**
> The service catalog is implemented as an extension API server and a controller. After Alibaba Cloud Container Service installs the service catalog function, the namespace catalog is created.



5. Click Clusters in the left-side navigation pane. Click Dashboard at the right of a cluster.



6. In the Kubernetes dashboard, select `catalog` as the Namespace in the left-side navigation pane. You can see the resource objects related to catalog apiserver and controller are installed under this namespace.



What's next

Then, you have successfully enabled the service catalog function. You can create a managed service by using the service broker in the service catalog, and apply the managed service to your applications.

18 Auto Scaling

18.1 Use an HPA auto scaling container

Alibaba Cloud Container Service supports the rapid creation of HPA-enabled applications on the console interface to achieve auto scaling of container resources. You can also configure it by defining the yaml configuration of Horizontal Pod Autoscaling (HPA).

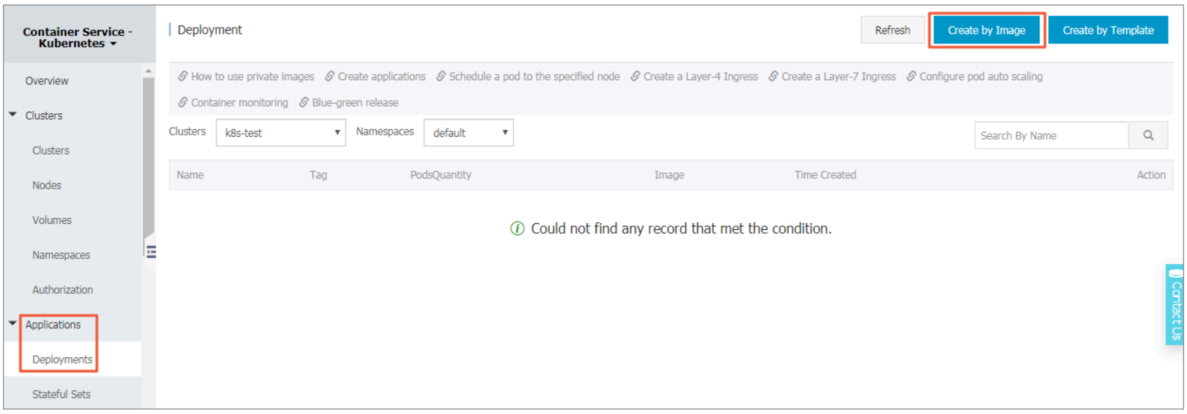
Prerequisites

- You have created a Kubernetes cluster. For more information, see Create a Kubernetes cluster.
- You have successfully connected to the master node of the Kubernetes cluster.

Method 1: Create an HPA application in the Container Service console

In Alibaba Cloud Container Service, HPA has been integrated. You can easily create it through the Container Service console.

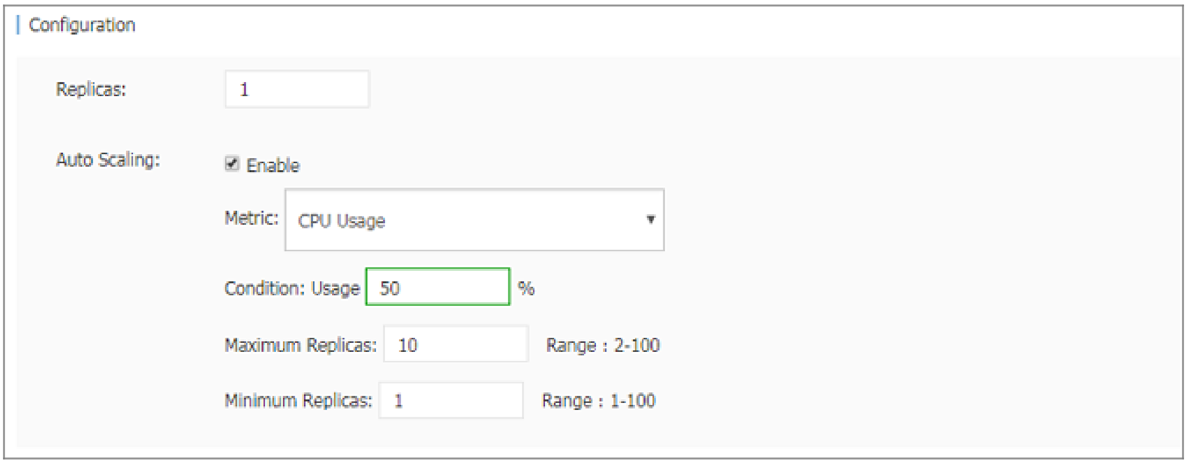
1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments. Then, click Create by Image in the upper-right corner.



3. Enter the application name, select the cluster and namespace, and click Next.

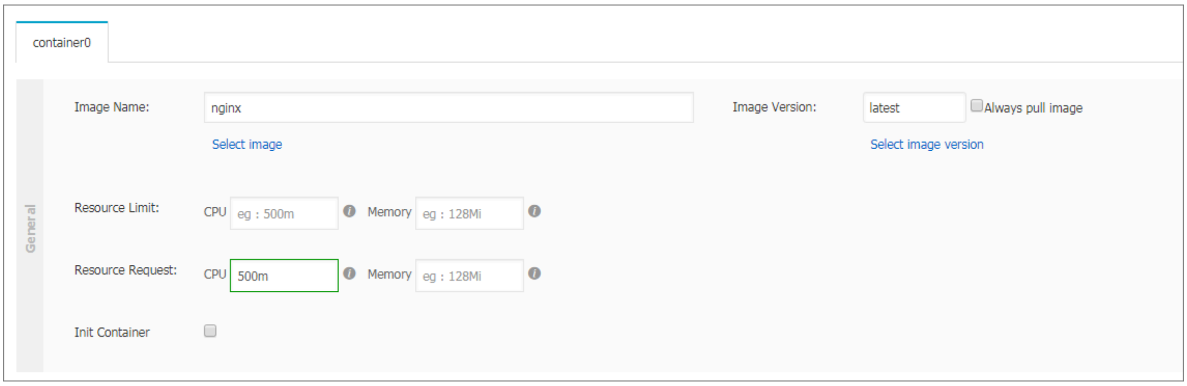
4. Configure the application settings. Set the number of replicas, select the Enable box for Automatic Scaling, and configure the settings for scaling.

    - Metric: CPU and memory. Configure a resource type as needed.
    - Condition: The percentage value of resource usage. The container begins to expand when the resource usage exceeds this value.
    - Maximum Replicas: The maximum number of replicas that the deployment can expand to.
    - Minimum Replicas: The minimum number of replicas that the deployment can contract to.



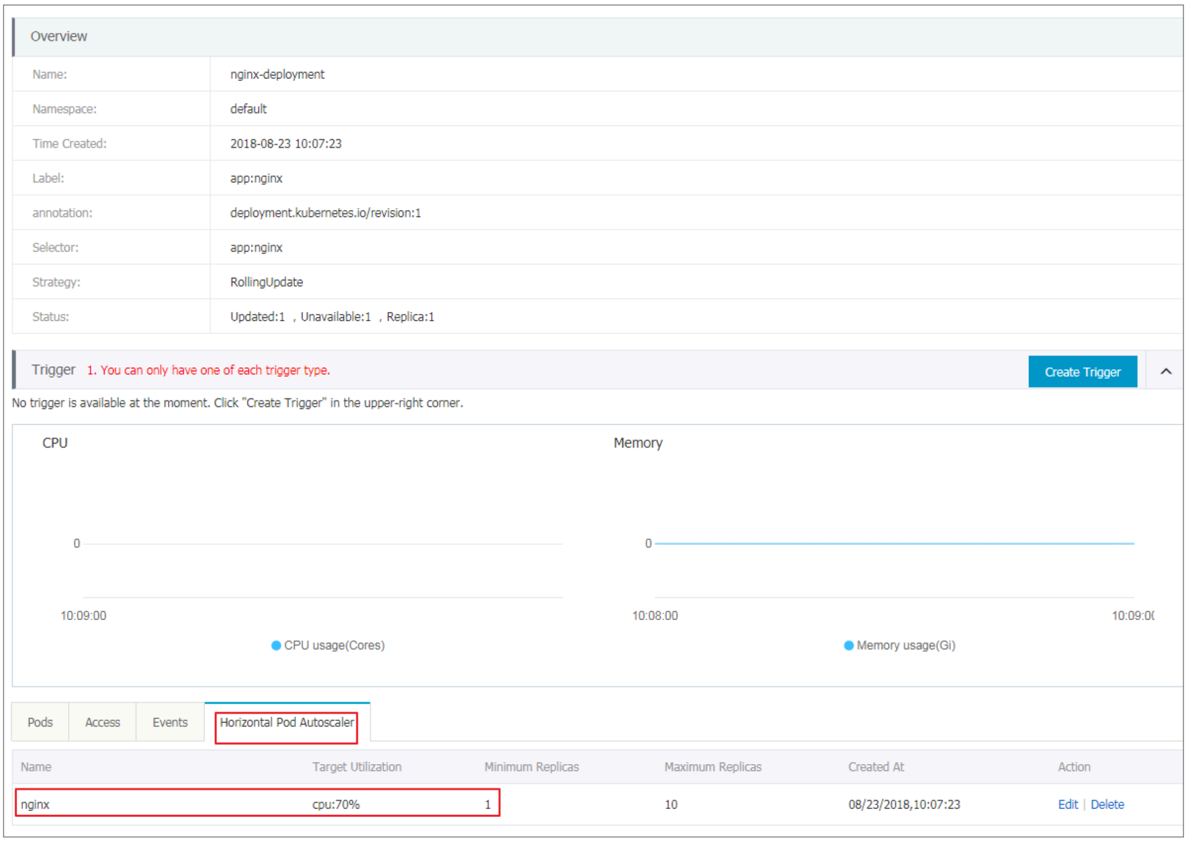
5. Configure the container. Select an image and configure the required resources. Click Next.

> **Note:**
> You must configure the required resources for the deployment. Otherwise, container auto scaling cannot be achieved.



6. In the Access Control page, do not configure any settings in this example. Click Create directly.

   Now a deployment that supports HPA has been created. You can view the auto scaling group information in the details of your deployment.



7. In the actual environment, the application scales according to the CPU load. You can also verify auto scaling in the test environment. By performing a CPU pressure test on the pod, you can find that the pod can complete the horizontal expansion in half a minute.

Method 2: Use kubectl commands to configure container auto scaling

You can also manually create an HPA by using an orchestration template and bind it to the deployment object to be scaled. Use the `kubectl` command to complete the container auto scaling configuration.

The following is an example of an Nginx application. Execute the `kubectl create -f xxx . yml` command to create an orchestration template for the deployment as follows:

```
apiVersion : apps / v1beta2 # for versions before 1 . 8 . 0
  use apps / v1beta1
kind : Deployment
metadata :
  name : nginx
  labels :
    app : nginx
spec :
  replicas : 2
  selector :
    matchLabel s :
      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 :
       - containerP ort : 80
        resources :
          requests :                     ## This parameter
must be configured . Otherwise , the HPA cannot operate .
            cpu : 500m
```

Create an HPA. Configure an object to which the current HPA is bound by using `scaleTarge tRef` . In this example, the object is the deployment named nginx.

```
apiVersion : autoscalin g / v2beta1
kind : Horizontal PodAutosca ler
metadata :
  name : nginx - hpa
  namespace : default
spec :
  scaleTarge tRef :                          ## Bind the HPA
  to a deployment named nginx
    apiVersion : apps / v1beta2
    kind : Deployment
    name : nginx
  minReplica s : 1
  maxReplica s : 10
  metrics :
 - type : Resource
    resource :
      name : cpu
      targetAver ageUtiliza tion : 50
```

Note:

> The HPA needs to configure the request resource for the pod.The HPA does not operate without the request resource.

Warnings similar to the following are displayed when you execute `kubectl describe hpa [ name ]`:

```
Warning    FailedGetR  esourceMet  ric         2m  ( x6   over   4m )
  horizontal - pod - autoscaler    missing   request   for   cpu   on
  container   nginx   in   pod   default / nginx - deployment - basic
- 75675f5897 - mqzs7

Warning    FailedComp  uteMetrics  Replicas    2m  ( x6   over
4m )   horizontal - pod - autoscaler    failed   to   get   cpu
utilizatio  n :  missing   request   for   cpu   on   container
nginx   in   pod   default / nginx - deployment - basic - 75675f5
```

After creating the HPA, execute the `kubectl describe hpa [ name ]` command again. You can see the following message, which indicates that the HPA is running normally.

```
Normal   Successful  Rescale   39s   horizontal - pod - autoscaler
New   size :  1 ;  reason :  All   metrics   below   target
```

When the usage of Nginx pod exceeds 50% set in this example, the container expands horizontally. When the usage of Nginx pod drops below 50%, the container contracts.

18.2 Autoscale the nodes of a Kubernetes cluster

This topic describes how to autoscale the nodes of a Kubernetes cluster to meet the requirements of your Kubernetes cluster workload. Alibaba Cloud Container Service for Kubernetes (ACK) provides the capability to autoscale the nodes of a Kubernetes cluster through using the cluster autoscaler program.

Background information

You can set the the cluster autoscaler to add different ECS instance types to your Kubernetes cluster, such as the general, GPU, and preemptive instance types. You can set multiple zones, instance specifications, and autoscaling modes.

Cluster autoscaler overview

The cluster autoscaler changes the size of a Kubernetes cluster based on the use of resource in the nodes of a pod in a Kubernetes cluster. Resource usage is calculated based on the pod resource requests.

When a pod requests more resources than what the associated node can provide, the pod enters the pending status. At which time, the autoscaler calculates the change to the size of cluster. It does so by calculating the number of nodes necessary to provide the requested resource with regard to the resource specification and threshold that you set for an autoscaling group.

For example, if you set a low threshold value for the number of nodes in an autoscalin g group, the cluster autoscaler deletes a node, which reduces the amount of resources that the pod can request.

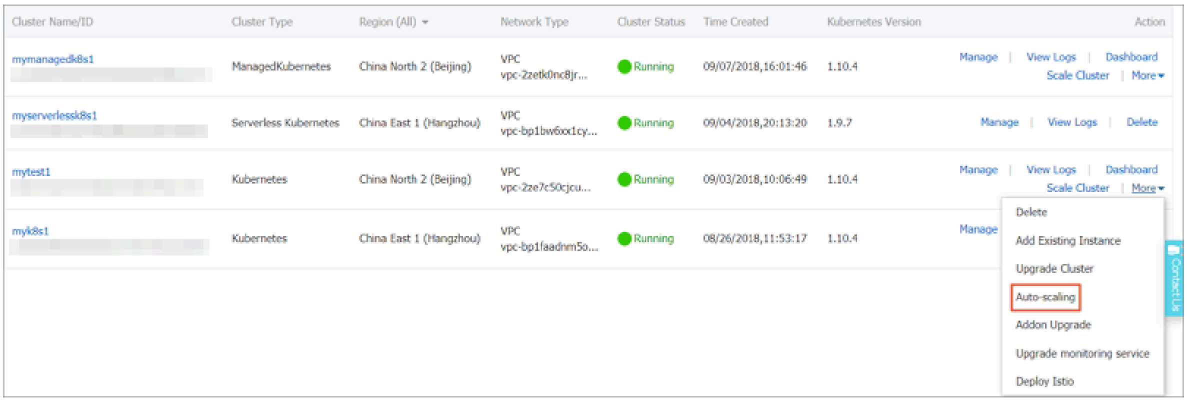
Notes

- By default, your account can use up to 30 Pay-As-You-Go ECS instances in all your clusters, and the route table of one VPC can contain up to 50 entries. To increase the number of available ECS instances or entries in a route table of one VPC, open a ticket.
- For a single type of ECS instances, the number of ECS instances of one specification that is permitted at one time varies frequently. Therefore, we recommend that you set multiple ECS instance types of one ECS specification.
- When a node for which you set the fast scaling mode is shut down and reclaimed, it is in the `NotReady` status. When the node is reused by the cluster autoscaler, the node enters the `Ready` status.
- When a node for which you set the fast scaling mode is shut down and reclaimed , only the disks attached to the node are charged (except that the node uses local disks, for example, ecs.d1ne.2xlarge).

Enable cluster autoscaling

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Clusters.

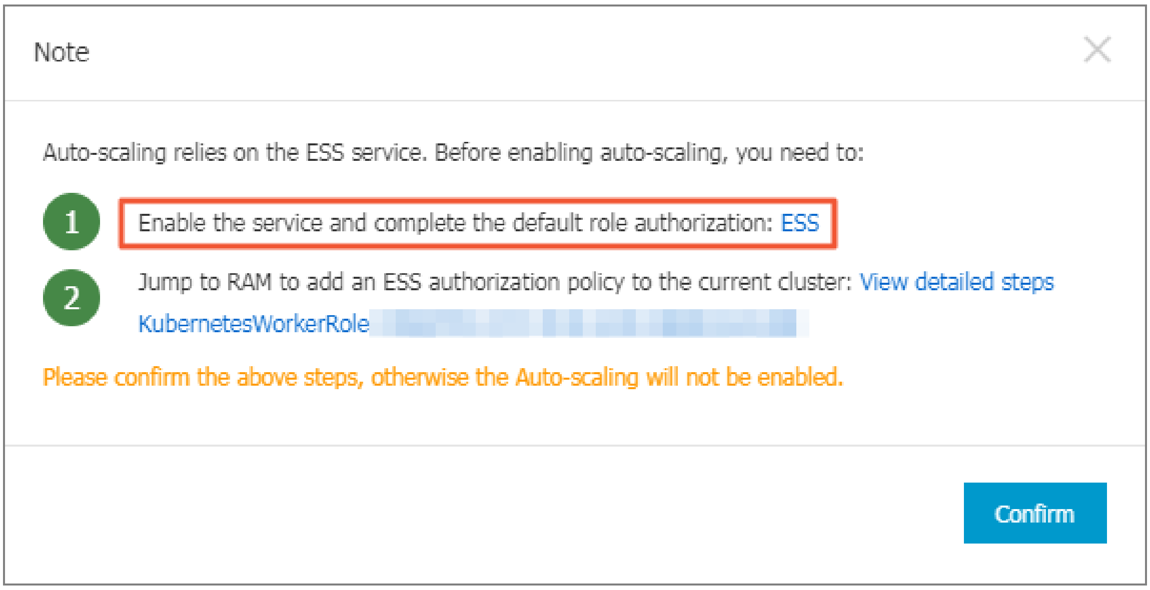
3. Find the target cluster. Then, in the Action column, choose More > Auto Scaling.



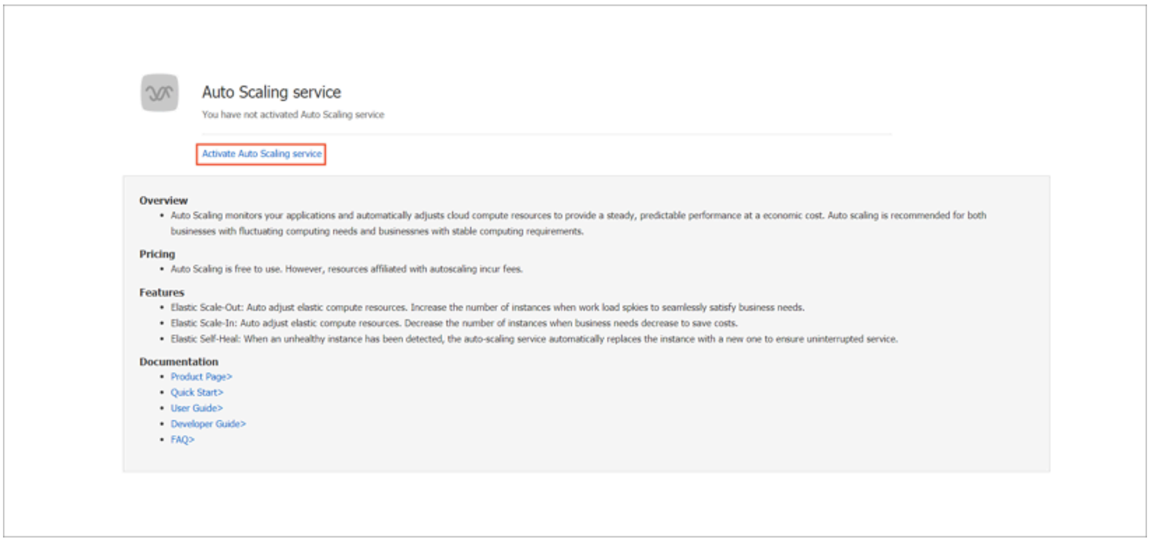
Grant required permissions for the Auto Scaling service and the cluster

- Activate the Auto Scaling service

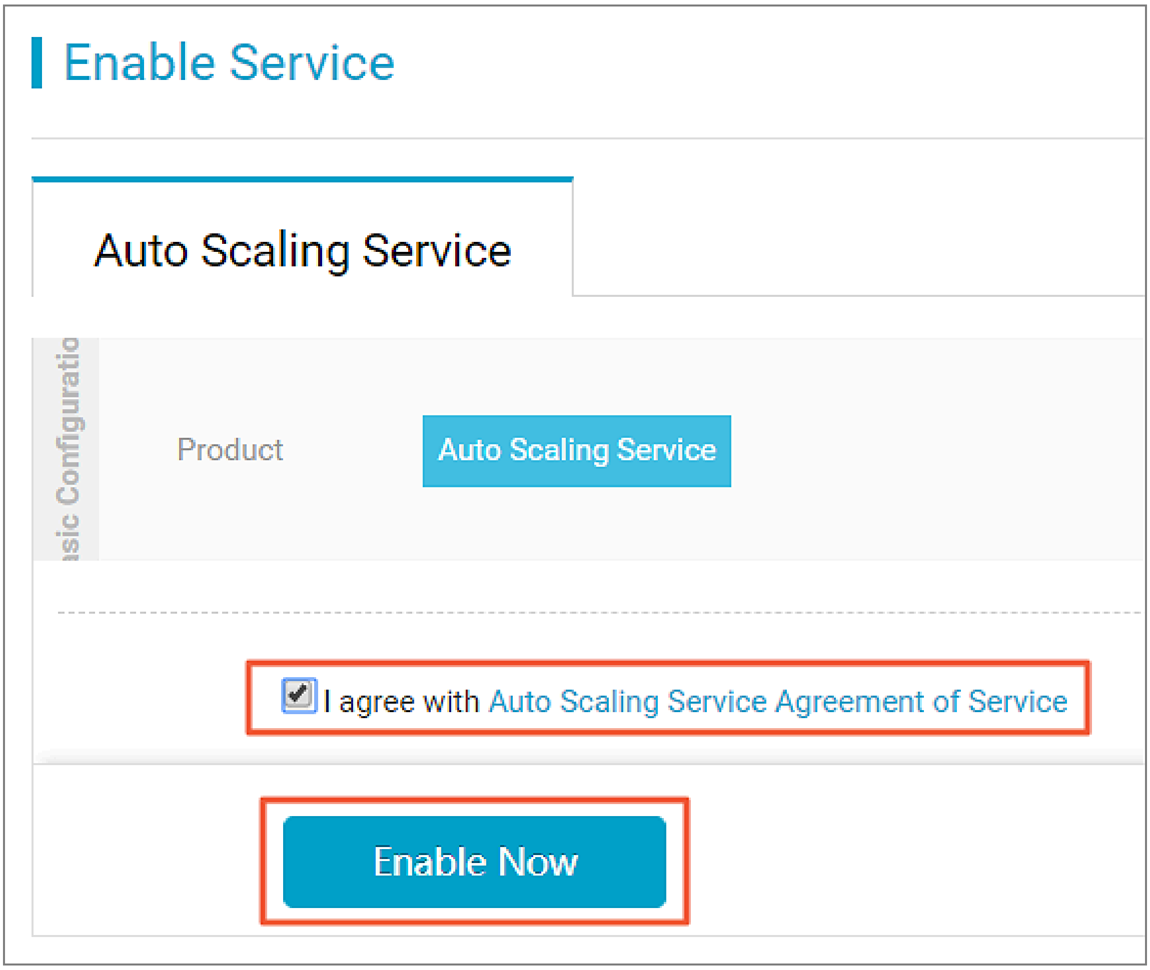
1. Click ESS in the displayed dialog box.



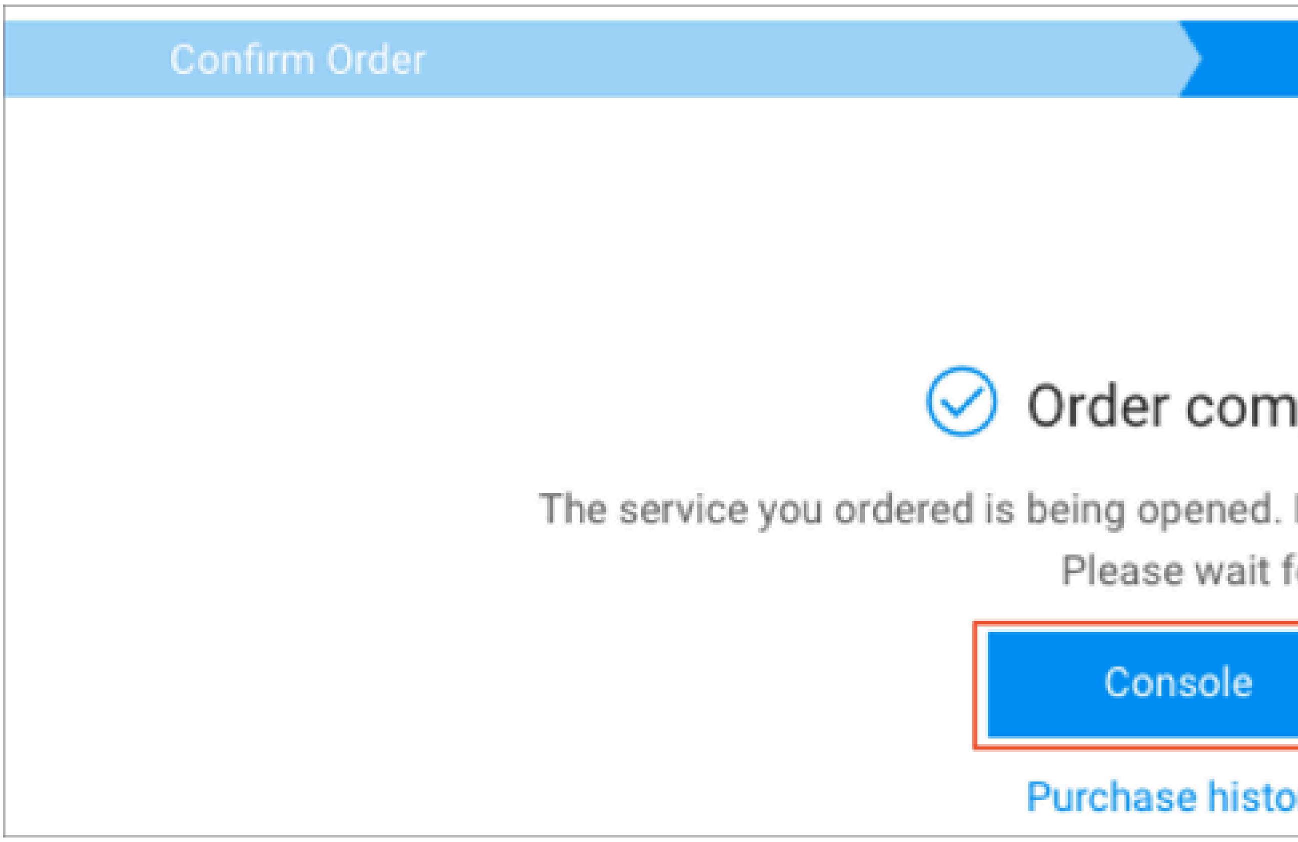
2. Click Activate Auto Scaling service.



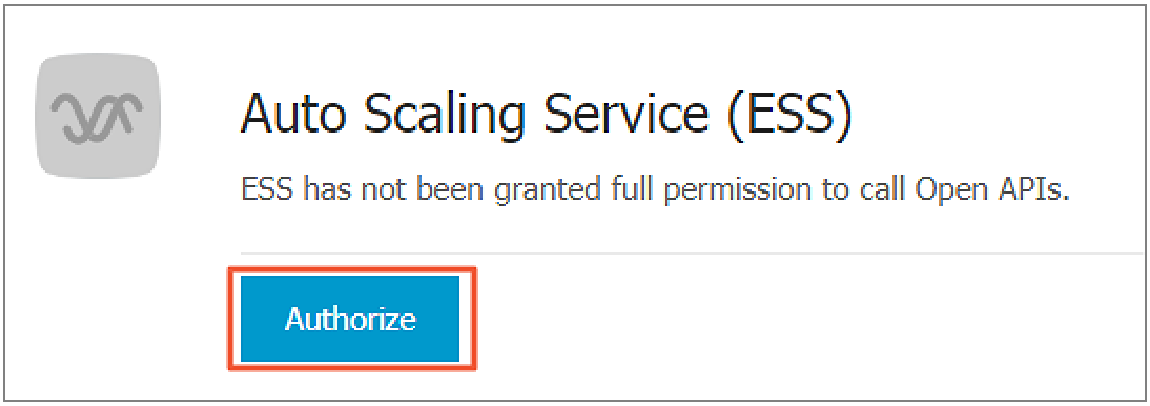
3. Read and confirm that you agree to the conditions by selecting the I agree with Auto Scaling Service Agreement of Service check box, and then click Enable Now.



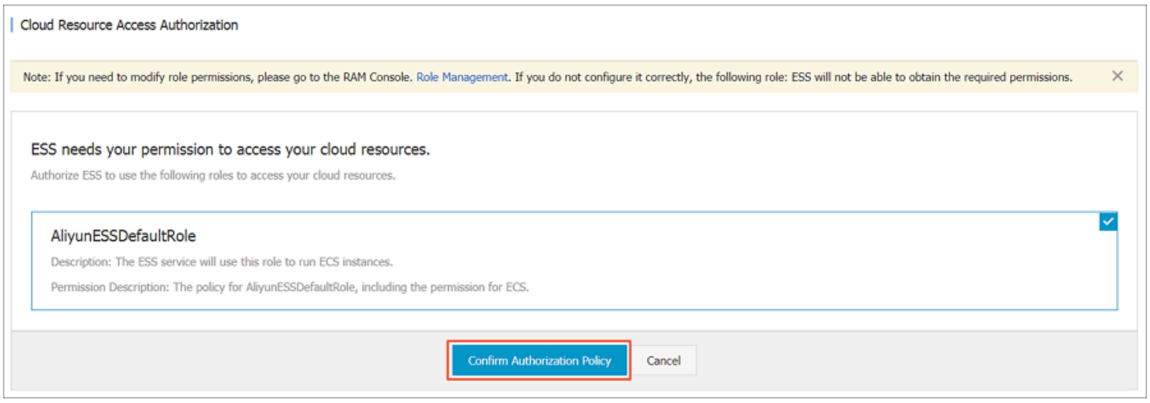
4. Click Console.



5. Click Authorize.



6. Click Confirm Authorization Policy to grant ESS the permission to access your cloud resources.



Verify the result

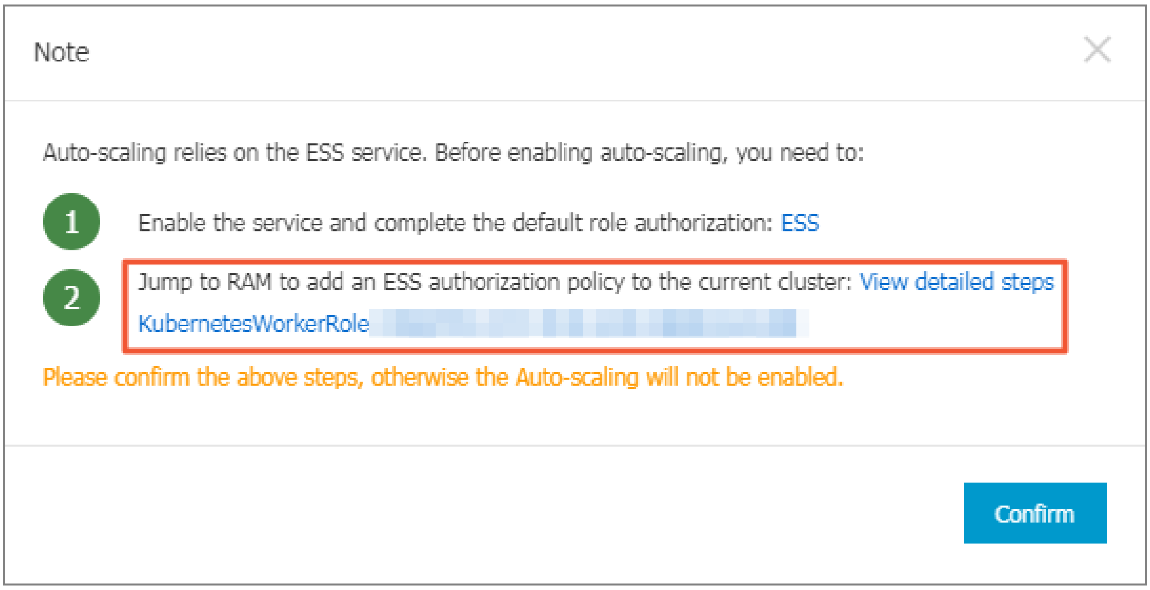
If the page automatically redirects to the Auto Scaling console, the activation is successful.

- Add ESS authorization policies to the cluster

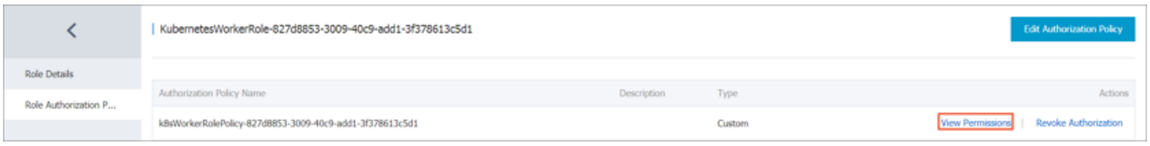
1. Click the Worker RAM role (`Kubernetes WorkerRole [ xxx ]`) in the following dialog box.

> **Note:**

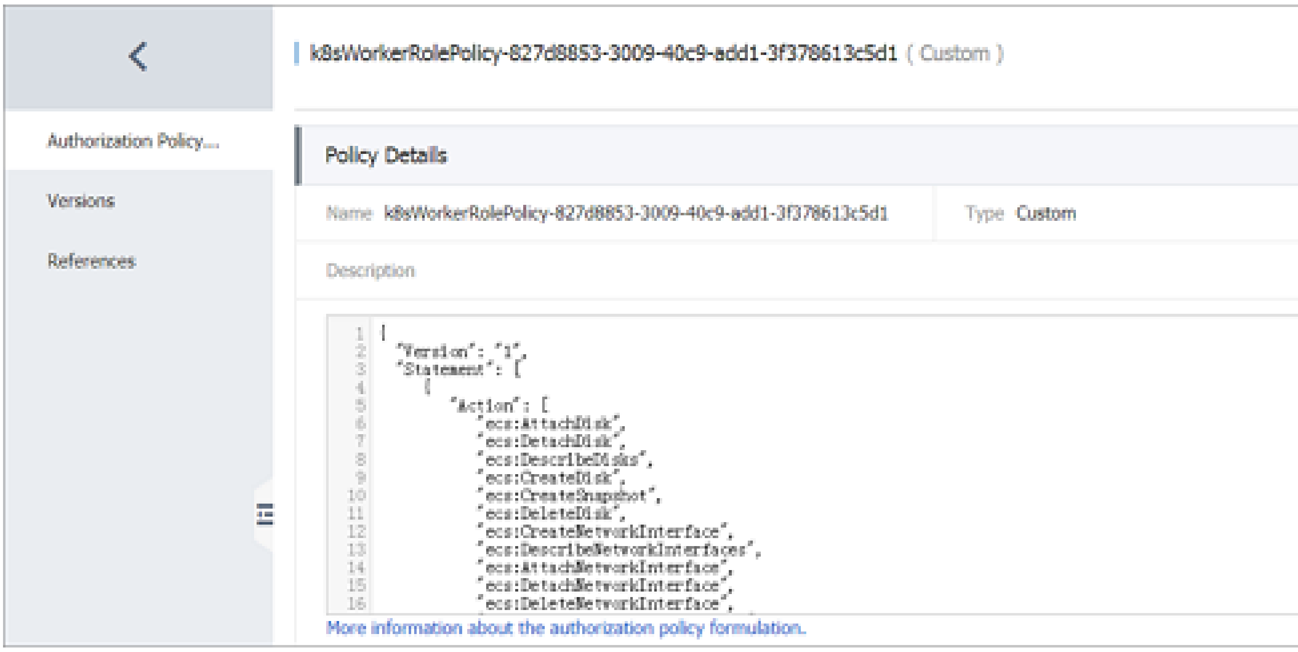
You need to use the primary account to log on to the console before perform this operation.



2. Click View Permissions on the right of the target authorization policy.



3. In the upper-right corner of the page, click Modify Authorization Policy.



4. In the `Action` field of the Policy Document area, add the following policies:

```
" ess : Describe *",
" ess : CreateScal  ingRule ",
" ess : ModifyScal  ingGroup ",
```

```
" ess : RemoveInst  ances ",
" ess : ExecuteSca  lingRule ",
" ess : ModifyScal  ingRule ",
" ess : DeleteScal  ingRule ",
" ecs : DescribeIn  stanceType  s ",
" ess : DetachInst  ances "
```

Note:

You must add a comma (,) to the end of the last line in the `Action` field before adding these policies.

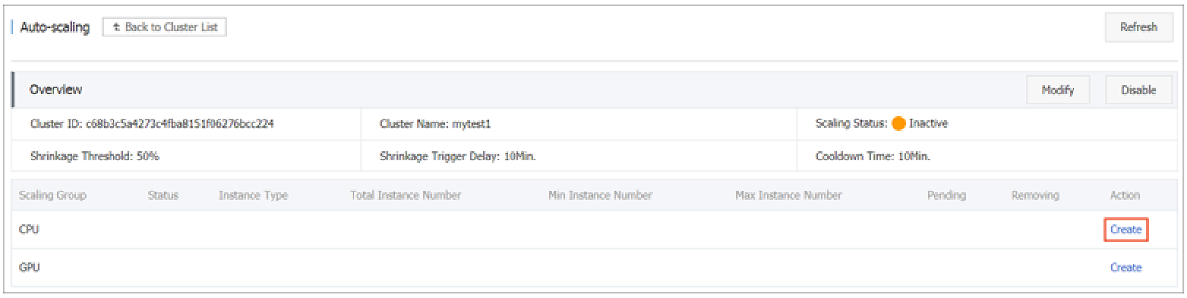
5. Click Modify Authorization.

Set cluster auto scaling parameters

1. On the Automatic Scaling page, set the following parameters.

Configuration	Description
Cluster	Target cluster name.
Scale-in Threshold	When the ratio of the amount of requested node resources to the total amount of node resources drops below this threshold, the system automatically scales in the cluster nodes. Note: After you enable the autoscaling feature for a Kubernetes cluster, the system automatically determines when to scale out the cluster nodes. Therefore, you only need to set the threshold used by the system to determine when to scale in the nodes of a cluster.
Defer Scale-in For	The number of minutes for which the system must wait to automatically scale in the cluster after the scale-in threshold is reached. The default value is 10 minutes.
Cooldown	The period (in minutes) during which the system does not automatically scale in or scale out a cluster after the number of cluster nodes increases or decreases. The default is 10 minutes.

2. Click Create on the right of the target type of resource (which can be CPU or GPU) that you want to autoscale.



On the Scaling Group Configuration page, set the following parameters to create a scaling group:

Configuration	Description
Region	The region to which the scaling group is deployed . You must ensure that the scaling group and the cluster where it is located share the same region. This region cannot be modified.
Zone	The zone where the scaling group is created.
VPC	The network where the scaling group is created. You must ensure that the scaling group and the cluster where it is located are in the same region.

Set worker nodes.

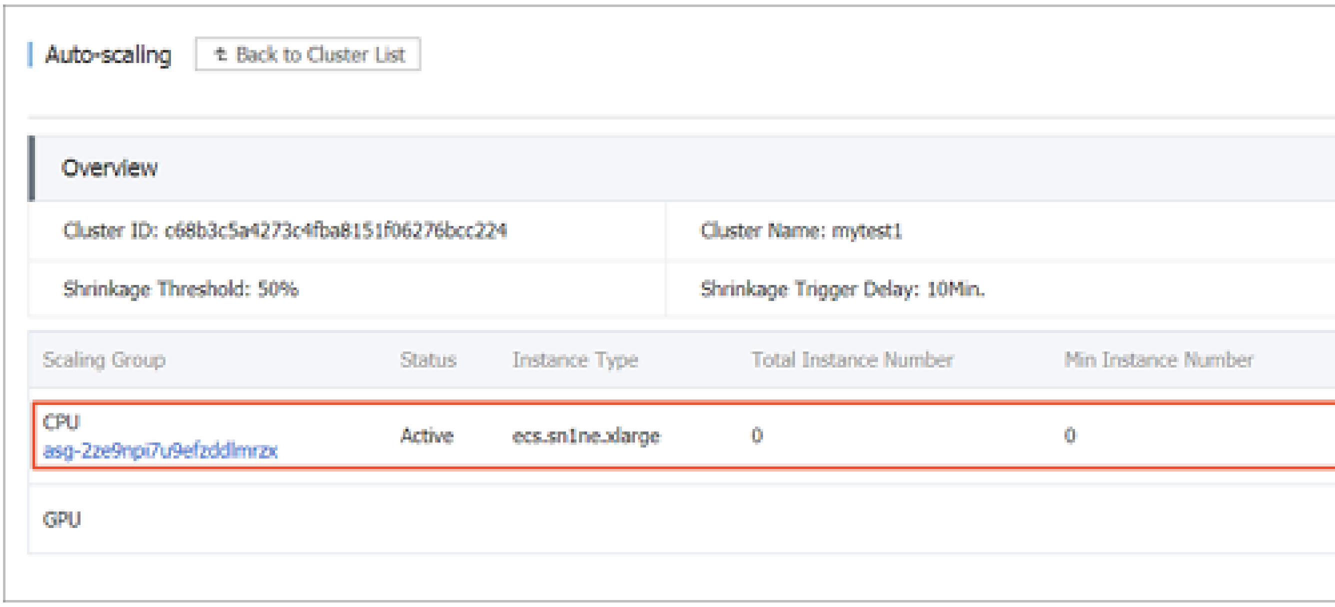
Configuration	Description
Instance Type	Set the specifications of instances in the scaling group .
System Disk	Set the system disks of the scaling group.
Attach Data Disk	Mount a data disk when you create a scaling group. By default, no data disk is mounted.
Instance Quantity	Set the number of instances in the scaling group. Note: · The number does not include the existing instances in the cluster. · By default, this parameter value is 0, and the cluster adds instances to the scaling group and the Kubernetes cluster where the scaling group is located when this parameter exceeds 0.

Configuration	Description
Key Pair	Set the key pair used to log on to the node added through autoscaling. You can create a new key pair in the Elastic Compute Service (ECS) console. **Note:** Only key pair logon is supported.
RDS Whitelist	Set the Relational Database Service (RDS) instances that can be accessed by the node added through autoscaling.

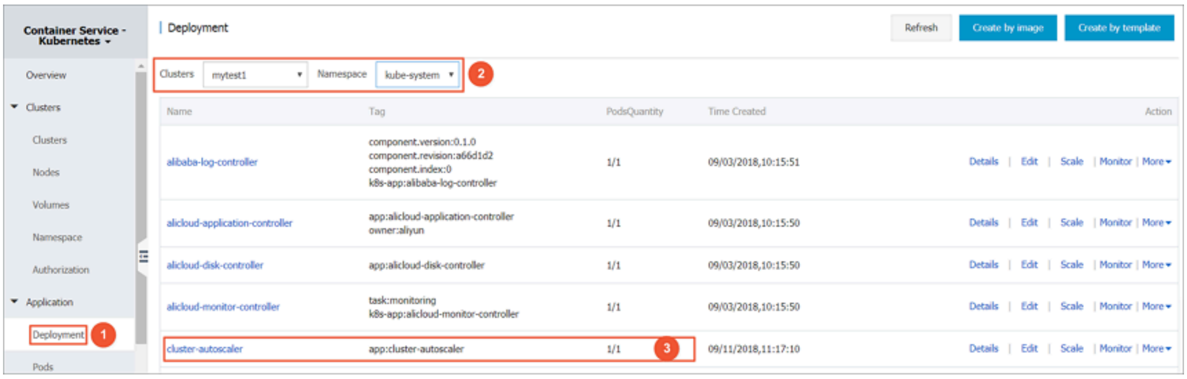
3. Click OK.

Verify the result

- You can directly verify that a scaling group under CPU is displayed on the Auto Scaling page.



- To verify the created autoscaling component, follow these steps:
  1. In the left-side navigation pane, choose Application > Deployment.
  2. Select the target cluster and the kube-system namespace to view the created component named cluster-autoscaler.



Troubleshooting

- If the cluster autoscaler cannot add nodes to a pod that requested more resources, you can perform the following checks:
  - Make sure that the amount of resources provided by the ECS instances that you set for the autoscaling group is greater than the amount of resources requested by the pod.
  - Make sure that you have granted the required permissions by following the preceding steps. You must grant the required permissions for each target cluster .
  - Make sure that the target Kubernetes cluster are connected to the Internet. The cluster autoscaler calls an API action from Alibaba Cloud, therefore you must ensure that the cluster nodes can be accessed through the Internet.

- If the cluster autoscaler cannot delete nodes from the autoscaling group, you can perform the following checks:
  - Make sure that the resource request threshold of pods on all nodes is not greater than that of used to scale in the cluster.
  - Make sure that no node run the pods that belong to the `kube - system` namespace.
  - Make sure that no node runs the pod for which any constrained scheduling policies are set. This is because a constrained scheduling policy can limit a pod to a fixed node.
  - Make sure that no pod contains a `PodDisrupt ionBudget` object that has reached the minimum value allowed. For more information, see How do Disruption Budgets work. For more information, see How do Disruption Budgets work.

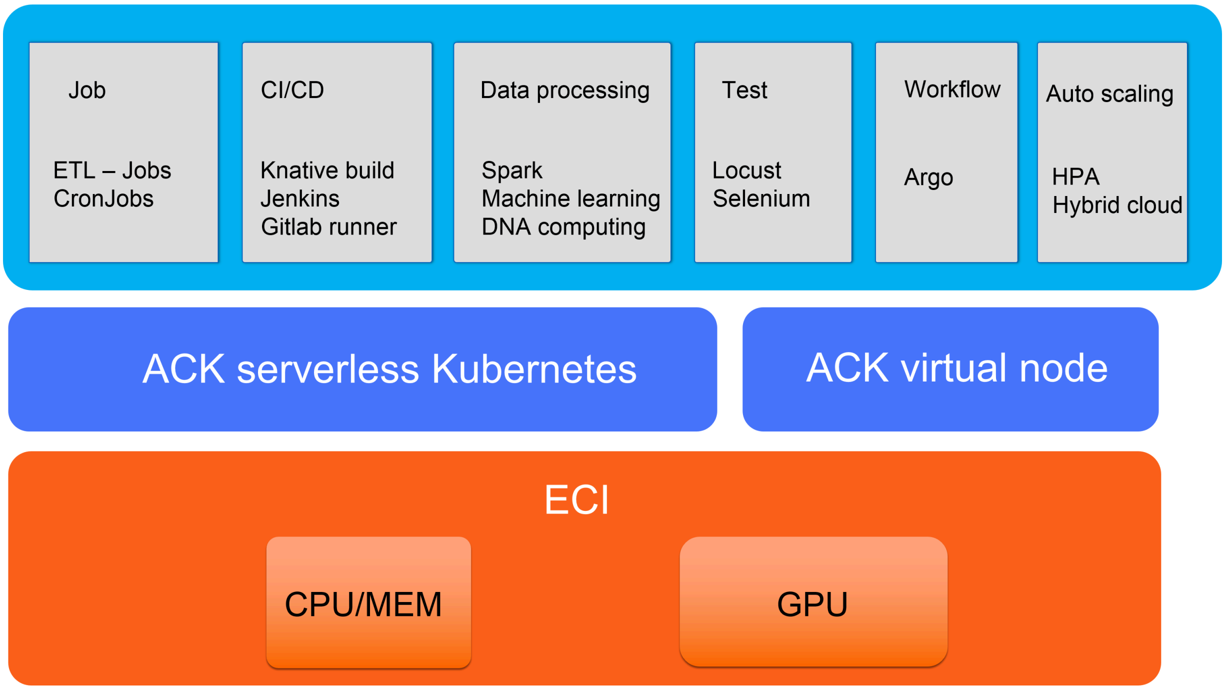
  For more information, see Cluster autoscaler.

18.3 Deploy a virtual node

This topic describes how to deploy a virtual node for a Kubernetes cluster by using the virtual node addon.

Background information

Virtual nodes are implemented by using Virtual Kubelet. Virtual nodes connect Kubernetes with Alibaba Cloud Elastic Container Instances (ECI), and provide Kubernetes clusters with a high level of elasticity. For more information about Virtual Kubelet, see How does Virtual Kubelet work?



Virtual nodes can help to enhance the elasticity of Kubernetes clusters. Specifically, virtual nodes based on Alibaba Cloud ECI can enhance the elasticity of clusters because they support GPU container instances, EIP addresses, and container instances of high specifications. Given the expansion capability provided by virtual nodes, you can easily manage multiple workloads for computing in a Kubernetes cluster.

In a cluster that contains virtual nodes, the pod in a physical node communicates with the ECI pod in the corresponding virtual node.

Note:

- The ECI pod in a virtual node is charged according to the specific amount of resources that you use. For information about ECI billing rules, see Billing overview.
- ECS instances of the specifications range of 0.25 vCPU to 64 vCPU are supported. For more information, see Limits.

Prerequisites

A managed Kubernetes cluster is created. For more information, see Create a managed Kubernetes cluster.

Create a virtual node for a cluster Work node

1. Log on to the Container Service console.

2. In the left-side navigation pane under Container Service-Kubernetes, choose Store > App Catalog. Then, click ack-virtual-node.
3. Click the Values tab, and then set these parameters.

    - `ECI_VSWITCH` : Set the VSwitch ID.

      To obtain the ID of the VSwitch that is associated with a Worker node, follow these steps:

      a. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Nodes.
      b. Select the target cluster, and then click a Worker node.

      In the Configuration Information area, the VSwitch ID is displayed.
    - `ECI_SECURITY_GROUP` : Set the security group ID.

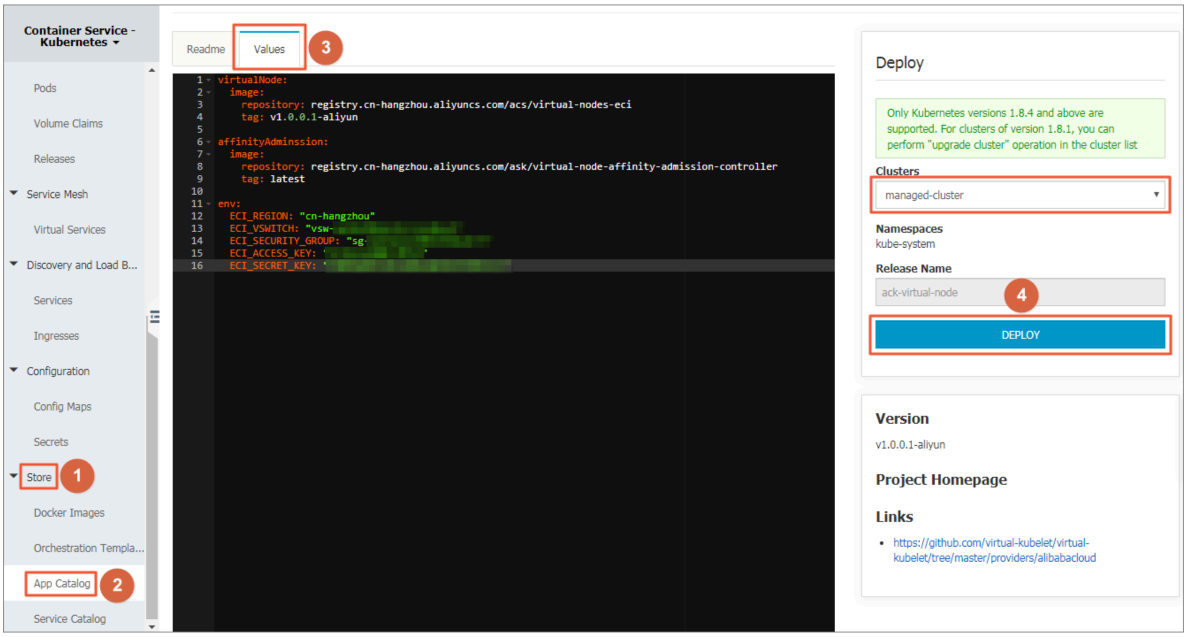
      To obtain the ID of the security group that is associated with a Worker node, follow these steps:

      a. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Nodes.
      b. Select the target cluster, and then click a Worker node.
      c. In the left-side navigation pane, click Security Groups.

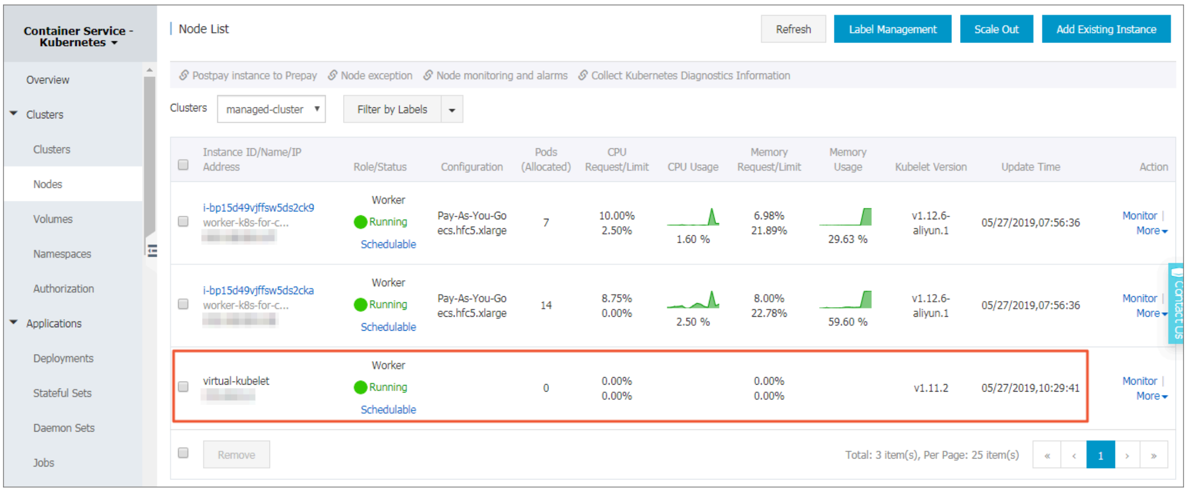
      On the Security Groups page, the security group ID is displayed.
    - `ECI_ACCESS_KEY` : Set the AccessKey.
    - `ECI_SECRET_KEY` : Set the SecretKey.
4. In the Deploy area on the right of the page, select the target cluster, and then click DEPLOY.

    Note:

By default, the namespace is set to `kube - system` , and the release name is set to `ack - virtual - node` .



5. In the left-side navigation pane under Container Service-Kubernetes, choose Clusters > Nodes to verify that a node named virtual-kubelet is displayed.



You can use kubectl commands to view cluster nodes, and the status of Helm deployment. You can also use Helm to upgrade and manage the deployed ack-virtual-node. For more information, see Use kubectl commands in Cloud Shell to manage a Kubernetes cluster.

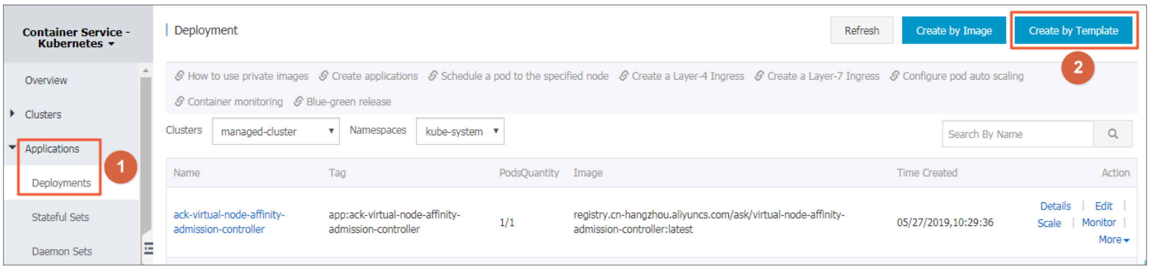
```
shell@Alicloud:~$ kubectl get no
NAME                                 STATUS   ROLES    AGE   VERSION
cn-hangzhou.i-bp14gdegum84fgei0lf4   Ready    <none>   16d   v1.12.6-aliyun.1
cn-hangzhou.i-bp14gdegum84fgei0lf5   Ready    <none>   16d   v1.12.6-aliyun.1
virtual-kubelet                      Ready    agent    72m   v1.11.2
shell@Alicloud:~$
shell@Alicloud:~$ helm ls
NAME                          REVISION   UPDATED                    STATUS     CHART                                 APP VERSION N
AMESPACE
ack-virtual-node-default      1          Fri Apr 12 08:10:46 2019   DEPLOYED   ack-virtual-node-v1.0.0.1-aliyun                  d
efault
```

Create a pod in the virtual node

If a Kubernetes cluster contains a virtual node, you can create a pod in the virtual node. Then, the virtual kubelet will create a corresponding ECI pod. You can use one of the following three methods to create a pod for the virtual node:

- Set `nodeSelect or` and `toleration s` to create a pod.

1. Log on to the Container Service console.
2. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments. Then, in the upper-right corner, click Create by Template.



3. Select the target cluster and namespace, select an example template or customize a template, and then click DEPLOY.

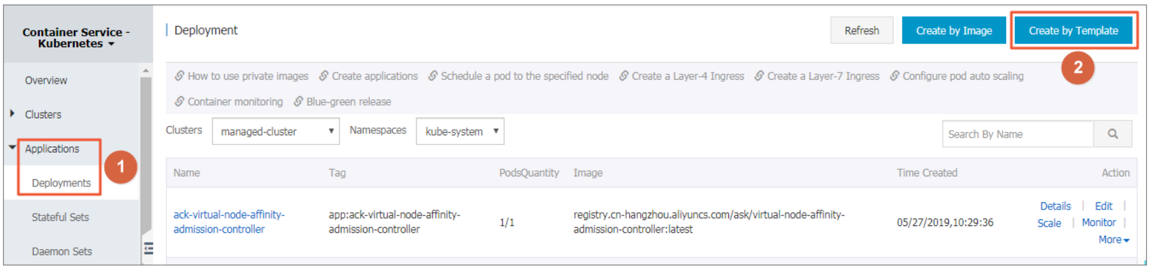
   You can use the following template to customize a pod:

```
apiVersion :  v1
kind :  Pod
metadata :
  name :  nginx
spec :
  containers :
 -  image :  nginx
    imagePullP  olicy :  Always
    name :  nginx
  nodeSelect  or :
    type :  virtual - kubelet
  toleration  s :
 -  key :  virtual - kubelet . io / provider
```

```
      operator :  Exists
```

- Set `nodeName` to create a pod.

1. In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Deployments. Then, in the upper-right corner, click Create by Template.



2. Select the target cluster and namespace, select an example template or customize a template, and then click DEPLOY.

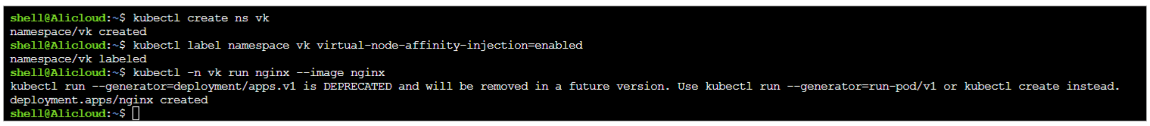
   You can use the following template to customize a pod:

```
apiVersion :  v1
kind :  Pod
metadata :
  name :  nginx
spec :
  containers :
 - image :  nginx
    imagePullP olicy :  Always
    name :  nginx
  nodeName :  virtual - kubelet
```

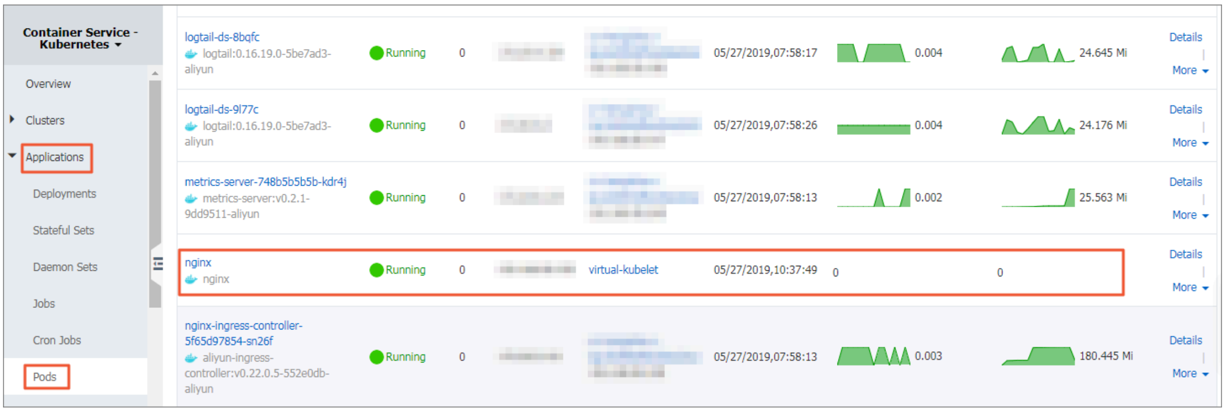
- Set a namespace tag to create a pod.

1. Use kubectl on Cloud Shell to connect to the target Kubernetes cluster. For more information, see [Use kubectl commands in Cloud Shell to manage a Kubernetes cluster](#).
2. Run the following commands to create a pod:

```
kubectl  create  ns  vk
kubectl  label  namespace  vk  virtual - node - affinity -
injection = enabled
kubectl  - n  vk  run  nginx  -- image  nginx
```



In the left-side navigation pane under Container Service-Kubernetes, choose Applications > Pods to verify that the pod is created.


Container Service - Kubernetes
Overview
Clusters
Applications
Deployments
Stateful Sets
Daemon Sets
Jobs
Cron Jobs
Pods
logtail-ds-8bqfc
logtail:0.16.19.0-5be7ad3-aliyun
Running
05/27/2019,07:58:17
0.004
24.645 Mi
logtail-ds-9l77c
logtail:0.16.19.0-5be7ad3-aliyun
05/27/2019,07:58:26
24.176 Mi
metrics-server-748b5b5b5b-kdr4j
metrics-server:v0.2.1-9dd9511-aliyun
05/27/2019,07:58:13
0.002
25.563 Mi
nginx
virtual-kubelet
05/27/2019,10:37:49
nginx-ingress-controller-5f65d97854-sn26f
aliyun-ingress-controller:v0.22.0.5-552e0db-aliyun
0.003
180.445 Mi
Details
More