Alibaba Cloud
Container Service for Kubernetes

User Guide

Issue: 20181217

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 information, supplementary instructions, and other content that the user must understand.	📋 **Note:** Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructio ns, 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]`*
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich` *`{stand	slave}`*

Contents

1 Kubernetes cluster

1.1 Introduction

1.1.1 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.

1.1.2 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.
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

Function	Description
	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.
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 capabiliti es. Supports upgrading a Kubernetes cluster to the latest version with one click.

Function	Description
	• Alibaba Cloud container team is responsibl e 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, orchestrat ion, and other technologies, which requires the devotion of professional personnel.
- The container technology is continuously developing with fast version iteration, which requires continuous upgrade and test.

1.2 Authorization

1.2.1 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.

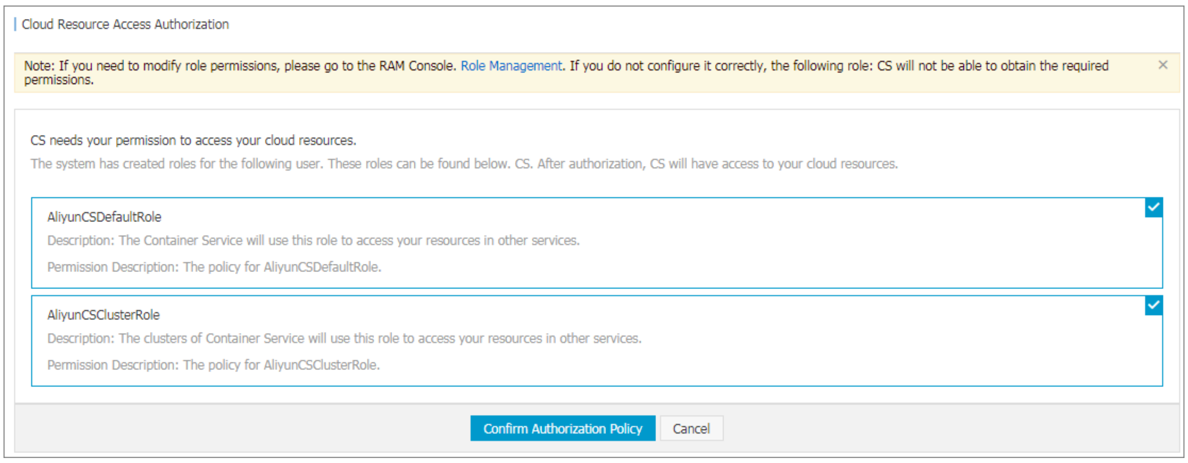
Instructions

- 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 sub-accounts*.

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**.


Cloud Resource Access Authorization

Note: If you need to modify role permissions, please go to the RAM Console. Role Management. If you do not configure it correctly, the following role: CS will not be able to obtain the required permissions.

CS needs your permission to access your cloud resources.
The system has created roles for the following user. These roles can be found below. CS. After authorization, CS will have access to your cloud resources.

AliyunCSDefaultRole
Description: The Container Service will use this role to access your resources in other services.
Permission Description: The policy for AliyunCSDefaultRole.

AliyunCSClusterRole
Description: The clusters of Container Service will use this role to access your resources in other services.
Permission Description: The policy for AliyunCSClusterRole.

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.
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.

Permission name (Action)	Permission description
vpc:AllocateEipAddress	Allocate EIP addresses.
vpc:AssociateEipAddress	Associate with EIP addresses.
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.

1.2.2 Use sub-accounts

Grant the sub-accounts the corresponding permissions before using the sub-accounts to log on to the Container Service console and perform the operations.

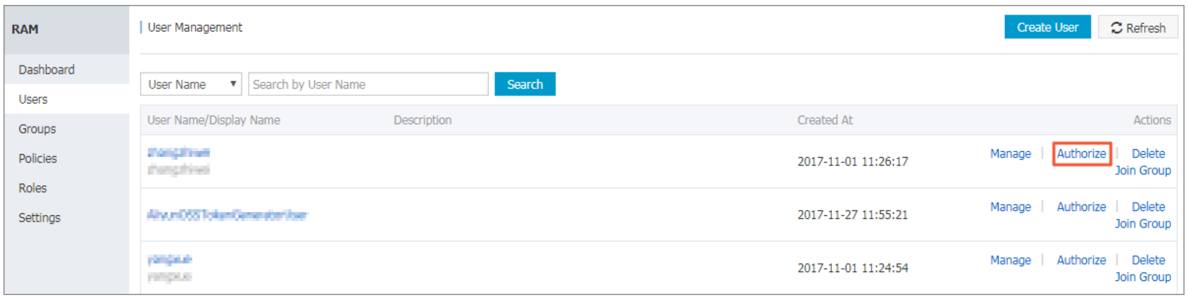
Step 1 Create sub-accounts and enable console logon

1. Log on to the *RAM console*.

2. Click **Users** in the left-side navigation pane. Click **Create User** in the upper-right corner.
3. Enter the username of the sub-account and then click **OK**.
4. On the User Management page, click **Manage** at the right of the created sub-account.
5. Click **Enable Console Logon** in the **Web Console Logon Management** section.
6. Enter the logon password in the appeared dialog box and click **OK**.

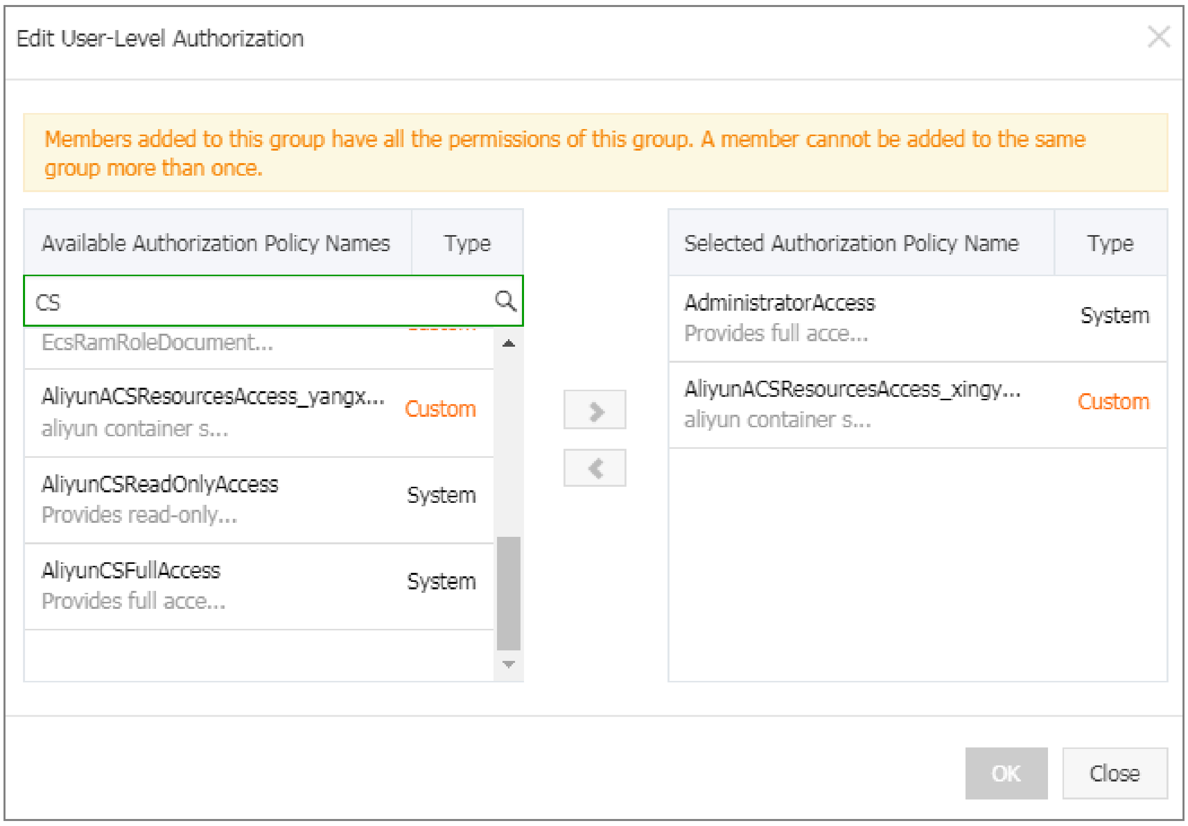
Step 2 Grant sub-accounts permissions to access Container Service

1. On the User Management page, click **Authorize** at the right of the created sub-account.



User Name/Display Name	Description	Created At	Actions
[redacted]		2017-11-01 11:26:17	Manage | Authorize | Delete Join Group
[redacted]		2017-11-27 11:55:21	Manage | Authorize | Delete Join Group
[redacted]		2017-11-01 11:24:54	Manage | Authorize | Delete Join Group

2. Select the authorization policy and click 1 to add the policy to the Selected Authorization Policy Name.



Edit User-Level Authorization

Members added to this group have all the permissions of this group. A member cannot be added to the same group more than once.

Available Authorization Policy Names	Type
CS	
EcsRamRoleDocument...	
AliyunACSResourcesAccess_yangx... aliyun container s...	Custom
AliyunCSReadOnlyAccess Provides read-only...	System
AliyunCSFullAccess Provides full acce...	System

Selected Authorization Policy Name	Type
AdministratorAccess Provides full acce...	System
AliyunACSResourcesAccess_xingy... aliyun container s...	Custom

You can use the following system default authorization policies:

- AliyunCSFullAccess: Provides full access to Container Service.

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

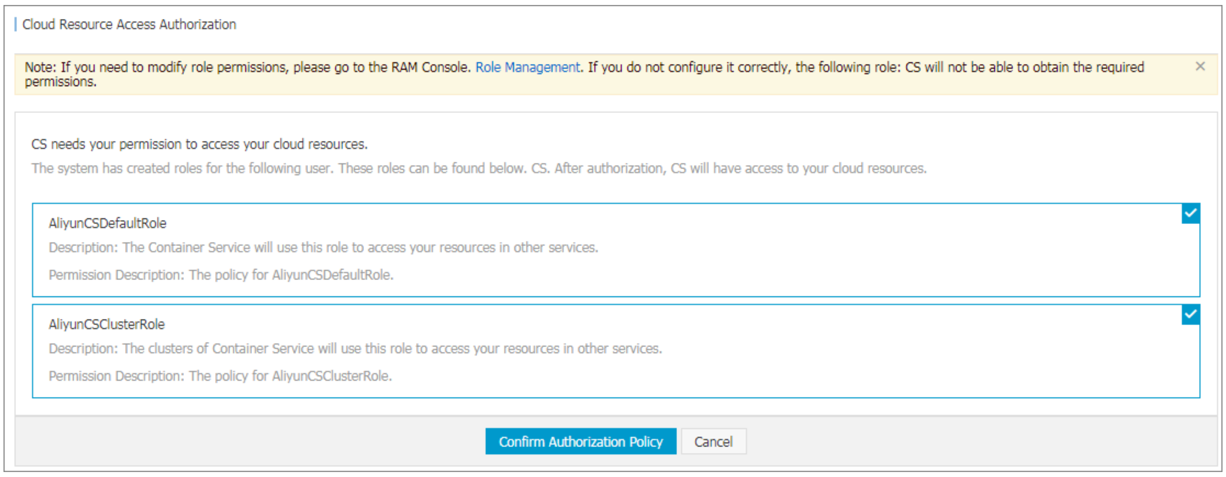
  You can also create custom authorization policies as per your needs and grant the policies to the sub-accounts. For more information, see *Create custom authorization policies*.

Step 3 Log on to Container Service console with sub-accounts

Log on to the *Container Service console* with a sub-account.

If you have granted the AliyunCSDefaultRole and AliyunCSClusterRole roles to the main account , you can use the sub-account directly to log on to the Container Service console and perform the operations.

If you have not granted the AliyunCSDefaultRole or AliyunCSClusterRole roles to the main account before, click **Confirm Authorization Policy** in the appeared Cloud Resource Access Authorization page.


Cloud Resource Access Authorization

Note: If you need to modify role permissions, please go to the RAM Console. Role Management. If you do not configure it correctly, the following role: CS will not be able to obtain the required permissions.

CS needs your permission to access your cloud resources.
The system has created roles for the following user. These roles can be found below. CS. After authorization, CS will have access to your cloud resources.

AliyunCSDefaultRole
Description: The Container Service will use this role to access your resources in other services.
Permission Description: The policy for AliyunCSDefaultRole.

AliyunCSClusterRole
Description: The clusters of Container Service will use this role to access your resources in other services.
Permission Description: The policy for AliyunCSClusterRole.

Then, refresh the Container Service console to perform the operations.

1.2.3 Create custom authorization policies

The authorization granularity of the system authorization policies provided by Container Service is coarse. If these authorization policies with coarse granularity 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 granularit y.

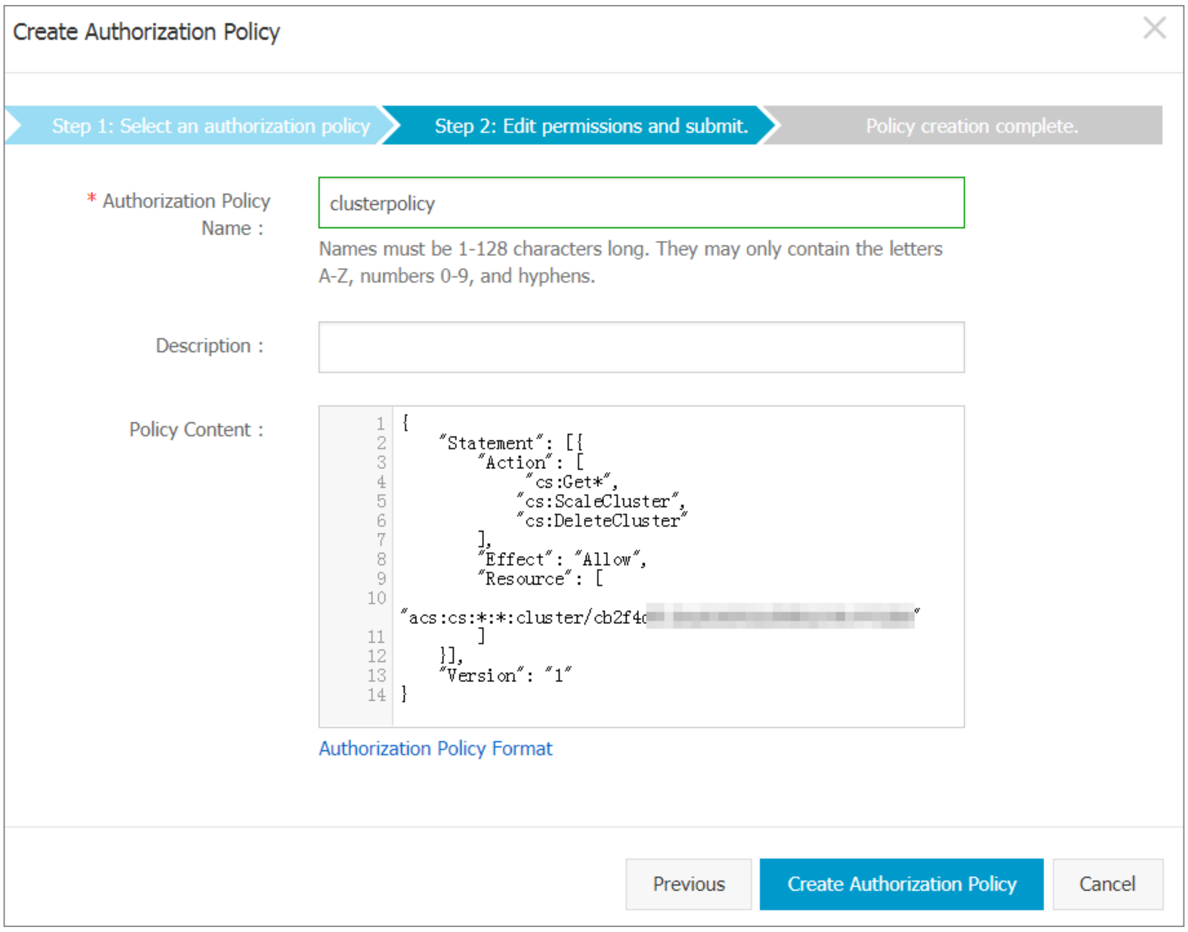
Create custom authorization policies

Get to know the basic structure and syntax of the authorization policy language before creating custom authorization policies. For more information, see *Authorization policy language descriptio ns*.

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.



Create Authorization Policy

Step 1: Select an authorization policy | Step 2: Edit permissions and submit. | Policy creation complete.

* Authorization Policy Name : clusterpolicy

Names must be 1-128 characters long. They may only contain the letters A-Z, numbers 0-9, and hyphens.

Description :

Policy Content :

```
{
    "Statement": [{
        "Action": [
            "cs:Get*",
            "cs:ScaleCluster",
            "cs:DeleteCluster"
        ],
        "Effect": "Allow",
        "Resource": [
"acs:cs:*:*:cluster/cb2f4c[redacted]"
        ]
    }],
    "Version": "1"
}
```

Authorization Policy Format

Previous | Create Authorization Policy | Cancel

```
{
 "Statement": [{
     "Action": [
         "cs:Get*",
         "cs:ScaleCluster",
         "cs:DeleteCluster"
     ],
     "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 1-1: 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.

Action	Description
GetClusterLogs	View cluster operation logs.
GetClusterEndpoint	View cluster access point.
GetClusterCerts	Download cluster certificate.
RevokeClusterCerts	Revoke cluster certificate.
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.

1.2.4 Sub-account Kubernetes permission configuration guide

This document helps you understand how to configure the Kubernetes Resource Access Management (RAM) cluster permissions for sub-accounts and the corresponding Kubernetes RBAC application permissions within the cluster through the Container Service console .

Prerequisite

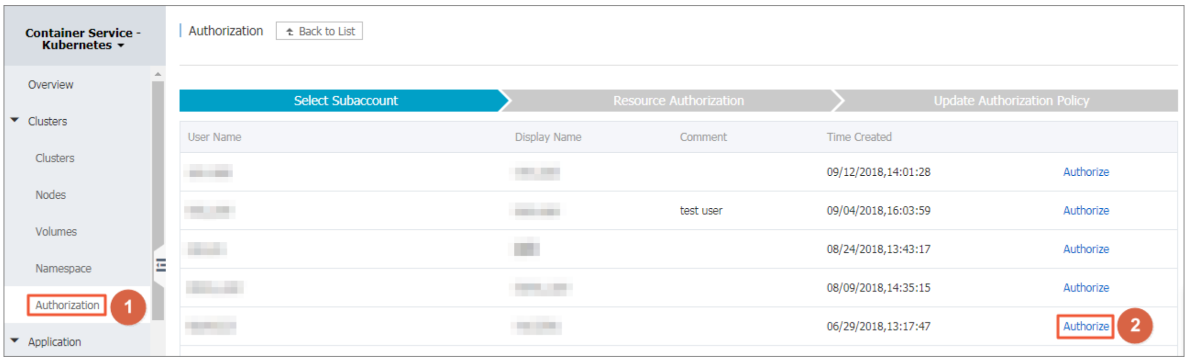
- The sub-account authorization page is visible only to the main account. You have an Alibaba Cloud main account and one or several sub-accounts.
- Due to the security restrictions of Alibaba Cloud RAM, when you modify the sub-account RAM authorization after you pass the authorization of the Container Service console, manually

perform authorization on the RAM console for the target sub-account according to the reference policy content and operation instructions on the page.

Procedure

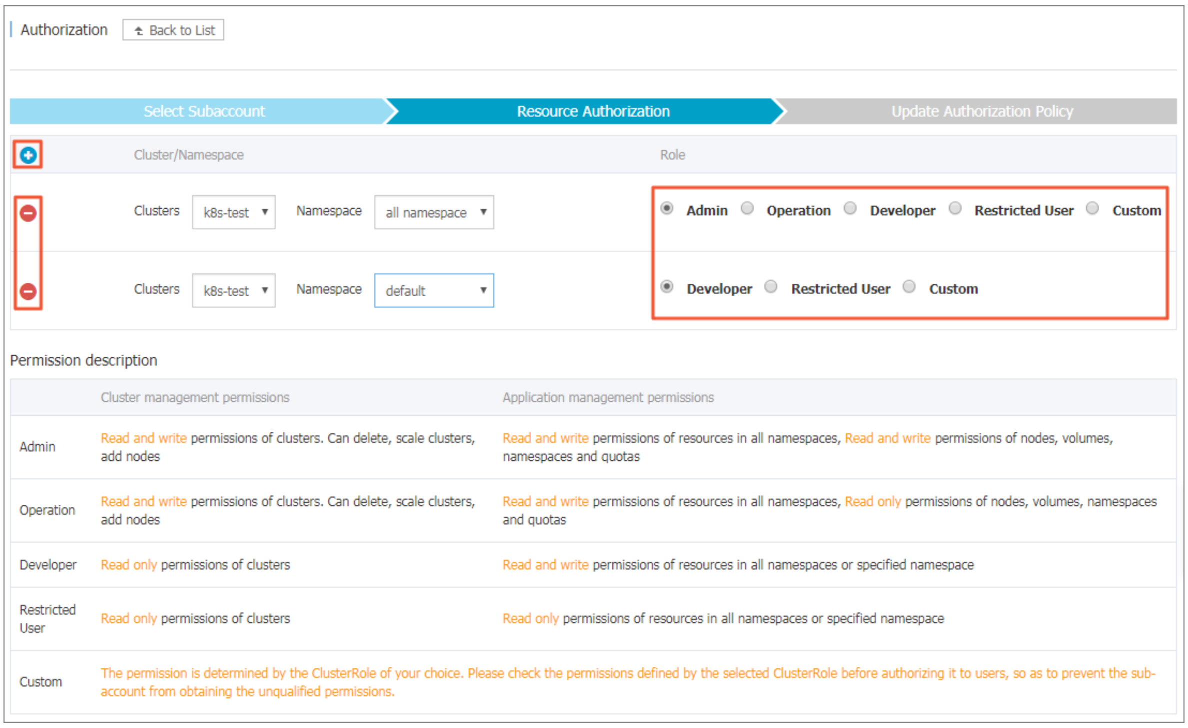
Note:
Use your main account to log on to the Container Service console because the sub-account authorization page is visible only to the main account.

1. Log on to the *Container Service console*.
2. Under the Kubernetes menu, click **Clusters** > **Authorization** in the left-side navigation pane.
3. In the sub-account list, select the sub-account that requires authorization, and click **Authorize**.



User Name	Display Name	Comment	Time Created	
[redacted]	[redacted]		09/12/2018,14:01:28	Authorize
[redacted]	[redacted]	test user	09/04/2018,16:03:59	Authorize
[redacted]	[redacted]		08/24/2018,13:43:17	Authorize
[redacted]	[redacted]		08/09/2018,14:35:15	Authorize
[redacted]	[redacted]		06/29/2018,13:17:47	Authorize

4. On the **Resource Authorization** page, you can add permission configurations of the cluster or namespace level by clicking the plus sign in the upper left corner of the table, and then select an role. You can also click the minus sign at the beginning of the configuration line to remove the target role.



Authorization ⬆ Back to List

Select Subaccount → Resource Authorization → Update Authorization Policy

Cluster/Namespace		Role
Clusters: k8s-test	Namespace: all namespace	[✓ Admin] Operation Developer Restricted User Custom
Clusters: k8s-test	Namespace: default	[✓ Developer] Restricted User Custom

Permission description

	Cluster management permissions	Application management permissions
Admin	Read and write permissions of clusters. Can delete, scale clusters, add nodes	Read and write permissions of resources in all namespaces, Read and write permissions of nodes, volumes, namespaces and quotas
Operation	Read and write permissions of clusters. Can delete, scale clusters, add nodes	Read and write permissions of resources in all namespaces, Read only permissions of nodes, volumes, namespaces and quotas
Developer	Read only permissions of clusters	Read and write permissions of resources in all namespaces or specified namespace
Restricted User	Read only permissions of clusters	Read only permissions of resources in all namespaces or specified namespace
Custom	The permission is determined by the ClusterRole of your choice. Please check the permissions defined by the selected ClusterRole before authorizing it to users, so as to prevent the sub-account from obtaining the unqualified permissions.	

For information about definitions of roles in clusters and namespaces, see the permission description below:

Table 1-2: Role permission description

	Cluster management permissions	Application management permissions
Admin	Read and write permissions of clusters. Delete clusters, scale clusters, and add nodes.	Read and write permissions of resources in all namespaces. Read and write permissions of nodes, volumes, namespaces, and quotas.
Operation	Read and write permissions of clusters. Delete clusters, scale clusters, and add nodes.	Read and write permissions of resources in all namespaces, Read only permissions of nodes, volumes, namespaces, and quotas.
Developer	Read only permissions of clusters.	Read and write permissions of resources in all namespaces or specified namespace.

	Cluster management permissions	**Application management permissions**
Restricted Users	Read only permissions of clusters.	Read only permissions of resources in all namespaces or specified namespace.
Custom	Read and write permissions of clusters. Delete clusters, scale clusters, and add nodes .	The permissions are determined by the ClusterRol e you select. Please confirm permissions that your selected ClusterRole have on resources before granting the permissions so as to avoid the condition that the sub-account obtains unexpected permissions.

5. After the configuration is completed, if you change the target sub-account RAM cluster permission, the Kubernetes cluster RAM permission reference configuration corresponding to the configuration item is displayed on the Update Authorization Policy page. You can complete the sub-account authorization update in the RAM console according to the instructions on the page.

Additional instructions

To avoid affecting the normal use of currently accessible Kubernetes clusters by the sub-accounts , the Container Service console is temporarily compatible with the old cluster access permission control. In a period of time, the original sub-account can access the Kubernetes cluster without RBAC application permission check. If you are a sub-account user, please contact the main account in time for authorization according to the cluster type and compatibility method below.

For the existing cluster created by the current sub-account, you can complete the automatic upgrade of the cluster application permissions by clicking **Upgrade current cluster authorization information** on the cluster details page.

After the end of the notice period, if a sub-account obtains no authentication from the main account or gets no permission management update, the sub-account is banned from accessing the application console corresponding to the cluster.

Table 1-3: Compatible cluster description

Compatible cluster type	Compatibility method
Clusters created by an existing sub-account	Prompt permission management upgrade notice and provide one-click upgrade link. The sub-account can complete application authorization by clicking the upgrade link.
Clusters with access authorized by an existing RAM	Prompt the permission management upgrade notice: Please contact the main account to complete the application authorization.
Clusters with access authorized by a newly created RAM	Prompt the permission management upgrade notice: Please contact the main account to complete the application authorization.

1.3 Cluster management

1.3.1 View cluster overview

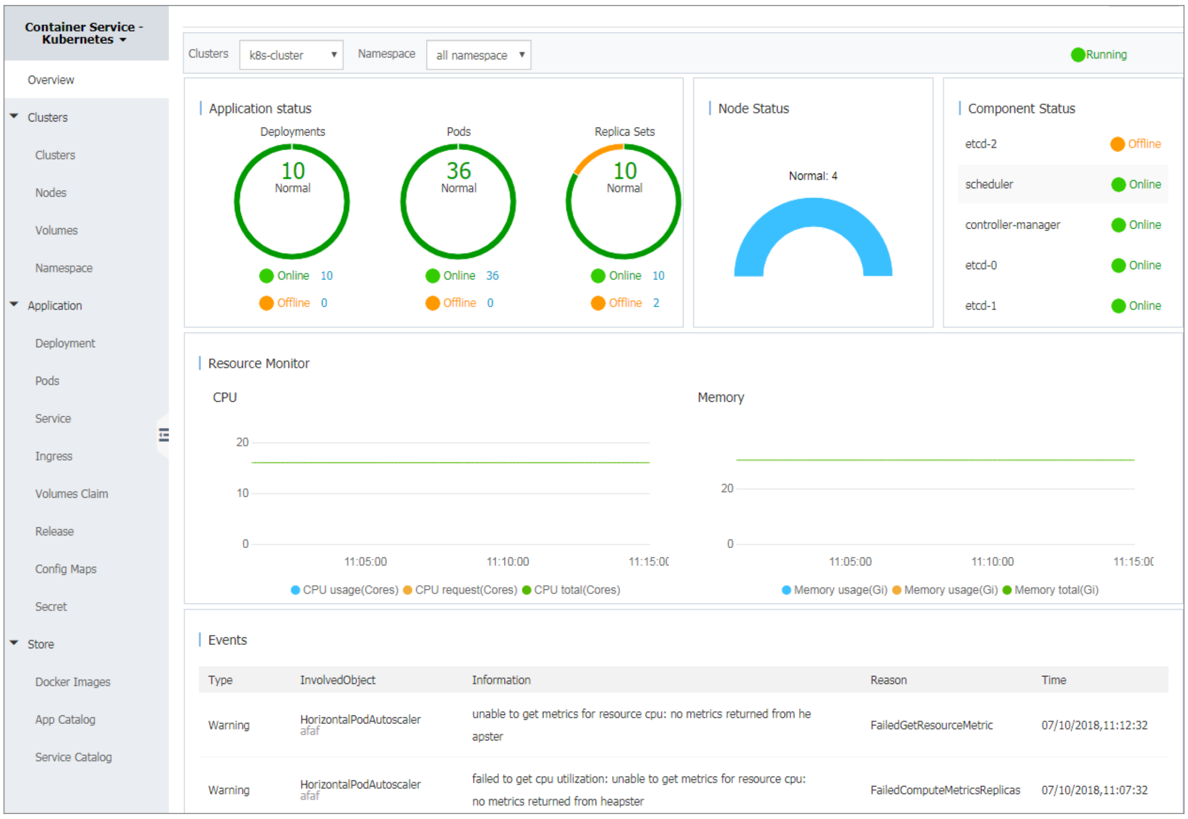
You can view the application status, component status, and resource monitoring charts on the Overview page of Alibaba Cloud Container Service Kubernetes clusters, which allows you to quickly understand the health status of clusters.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetesu, click **Overview** in the left navigation bar to enter the Kubernetes cluster overview page.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. You can view the application status, component status, and resource monitoring charts.
    - **Application status**: The status of deployments, pods, and replica sets that are currently running. Green indicates the normal status and orange indicates an exception.
    - **Node status**: Displays the node status of the current cluster.
    - **Component status**: The components of Kubernetes clusters are generally deployed under the kube-system namespace, including the core components such as scheduler, controller-manager, and etcd.
    - **Resource monitor**: Provides the monitoring charts of CPU and memory. CPU is measured in cores and is accurate to three decimal places. The minimum unit is millicores, that is, one

thousandth of one core. Memory is measured in G and is accurate to three decimal places. For more information, see *Meaning of CPU* and *Meaning of memory*.

- **Event**: Displays event information of the cluster, such as warnings and error events.



Component Status

etcd-2	Offline
scheduler	Online
controller-manager	Online
etcd-0	Online
etcd-1	Online

Events

Type	InvolvedObject	Information	Reason	Time
Warning	HorizontalPodAutoscaler afaf	unable to get metrics for resource cpu: no metrics returned from heapster	FailedGetResourceMetric	07/10/2018,11:12:32
Warning	HorizontalPodAutoscaler afaf	failed to get cpu utilization: unable to get metrics for resource cpu: no metrics returned from heapster	FailedComputeMetricsReplicas	07/10/2018,11:07:32

1.3.2 Create a Kubernetes cluster

You can create a Kubernetes cluster quickly and easily in the Container Service console.

Context

During cluster creation, Container Service performs the following operations:

- Create Elastic Compute Service (ECS) instances and configure to log on to other nodes from management nodes with the SSH public key. Install and configure the Kubernetes cluster by using CloudInit.
- Create a security group. This security group allows the Virtual Private Cloud (VPC) inbound access of all the ICMP ports.
- Create a new VPC and VSwitch if you do not use the existing VPC, and then create SNAT for the VSwitch.
- Create VPC routing rules.
- Create a NAT Gateway and a shared bandwidth package or Elastic IP (EIP).

- Create a Resource Access Management (RAM) user and an AccessKey. The RAM user has the permissions for querying, creating, and deleting ECS instances, the permissions for adding and deleting cloud disks, and all permissions for the operations on Server Load Balancer (SLB), CloudMonitor, VPC, Log Service, and Network Attached Storage (NAS). The Kubernetes cluster dynamically creates SLB instances, cloud disks, and VPC routing rules according to your configurations.
- Create an intranet SLB instance and expose the port 6443.
- Create an Internet SLB instance and expose the port 6443. (If you select to enable SSH access for Internet when creating the cluster, port 22 is exposed. Otherwise, port 22 is not exposed.)

Prerequisites

The services such as Container Service, Resource Orchestration Service (ROS), and RAM have been activated.

Log 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 instances created with the cluster support only the Pay-As-You-Go billing method.
- Kubernetes clusters support only the VPC network type.
- By default, each account has a specified quota for the cloud resources it can create. If the number of cloud resources exceeds the quota, the account cannot create a cluster. Make sure you have enough quota before creating a cluster. To increase your quota, open a ticket.
  - By default, each account can create up to 5 clusters in all regions and add up to 40 nodes to each cluster. To create more clusters or nodes, open a ticket.

    Note:
    In a Kubernetes cluster, the maximum number of default VPC routs is 48, that is, the Kubernetes cluster has up to 48 nodes by default when using VPC. To increase the

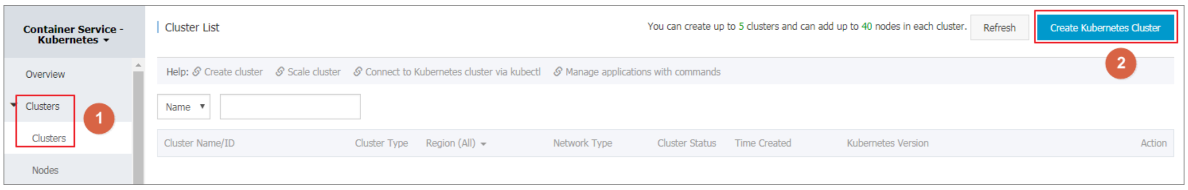
number of nodes, first open a ticket for the target VPC so as to increase the number of VPC routes, and then open a ticket for Container Service.

- 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.
  - The Pay-As-You-Go and Subscription ECS instances can be created.

Procedures

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

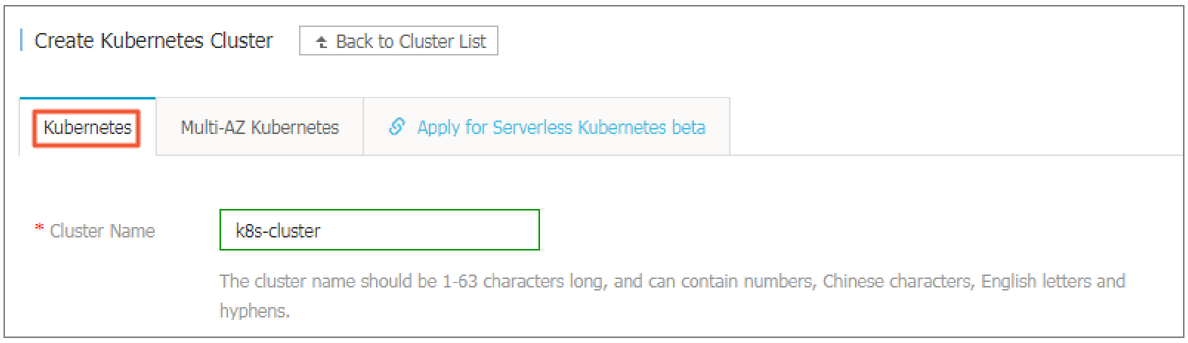


You can create up to 5 clusters and can add up to 40 nodes in each cluster.

Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Time Created	Kubernetes Version	Action

By default, the **Create Kubernetes Cluster** page is displayed.


Create Kubernetes Cluster
Back to Cluster List
Kubernetes
Multi-AZ Kubernetes
Apply for Serverless Kubernetes beta

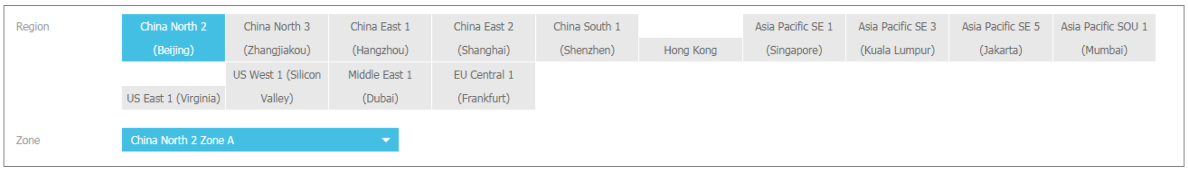
* Cluster Name: k8s-cluster

The cluster name should be 1-63 characters long, and can contain numbers, Chinese characters, English letters and hyphens.

4. Enter the cluster name.

   The cluster name can be 1–63 characters long and contain numbers, Chinese characters, English letters, and hyphens (-).

5. Select the region and zone where the cluster is located.



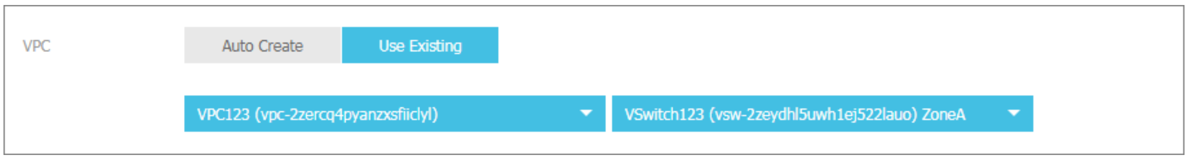
Zone: China North 2 Zone A

6. Set the cluster network type. Kubernetes clusters support only the VPC network type.

VPC: You can select **Auto Create** to create a VPC together with the Kubernetes cluster, or select**Use Existing** to use an existing VPC. If you select **Use Existing**, you can select a VPC and VSwitch from the two displayed drop-down lists.

- **Auto Create**: The system automatically creates a NAT Gateway for your VPC when a cluster is created.
- **Use Existing**: If the selected VPC has a NAT Gateway, Container Service uses the NAT Gateway. Otherwise, the system automatically creates a NAT Gateway by default. If you do not want the system to automatically create a NAT Gateway, deselect the **Configure SNAT for VPC** check box.

Note:
If you deselect the check box, configure the NAT Gateway on your own to implement the VPC Internet environment with secure access, or manually configure the SNAT. Otherwise, instances in the VPC cannot access the Internet normally, which leads to cluster creation failure.



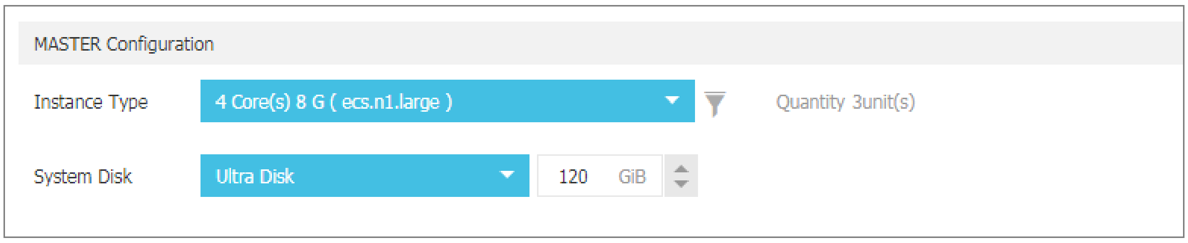
VPC123 (vpc-2zercq4pyanzxsfiiclyl)
VSwitch123 (vsw-2zeydhl5uwh1ej522lauo) ZoneA

7. Set the node type. Pay-As-You-Go and Subscription types are supported.
8. Configure the Master nodes.

   Select the instance type for the Master nodes.

Note:

- Currently, only the CentOS operating system is supported.
- Currently, you can create only three Master nodes.
- System disks are attached to the Master nodes by default. Available system disks are SSD Cloud Disks and Ultra Cloud Disks.

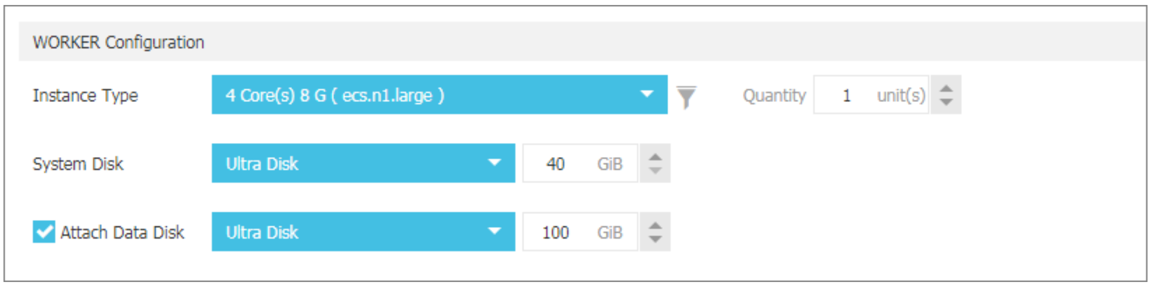

MASTER Configuration
Instance Type: 4 Core(s) 8 G (ecs.n1.large) Quantity 3unit(s)
System Disk: Ultra Disk 120 GiB

9. Configure the Worker nodes. You can create Worker nodes or add existing instances.

Note:

- Currently, only the CentOS operating system is supported.
- Each cluster can contain up to 37 Worker nodes. To create more nodes, open a ticket.
- System disks are attached to the Worker nodes by default. Available system disks are SSD Cloud Disks and Ultra Cloud Disks.
- You can also manually attach a data disk to a Worker node. The disk can be an SSD Cloud Disk, an Ultra Cloud Disk, or a Basic Disk.

a. To create Worker nodes, select the instance type and set the number of Worker nodes. In this example, create one Worker node.



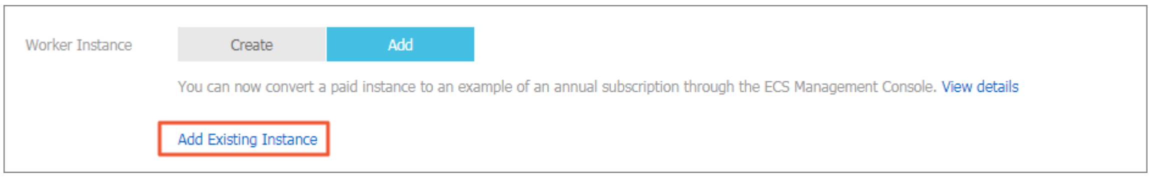
WORKER Configuration

Instance Type: 4 Core(s) 8 G (ecs.n1.large) Quantity 1 unit(s)

System Disk: Ultra Disk 40 GiB

✓ Attach Data Disk: Ultra Disk 100 GiB

b. To add existing instances, you must create ECS instances in the current region in advance.



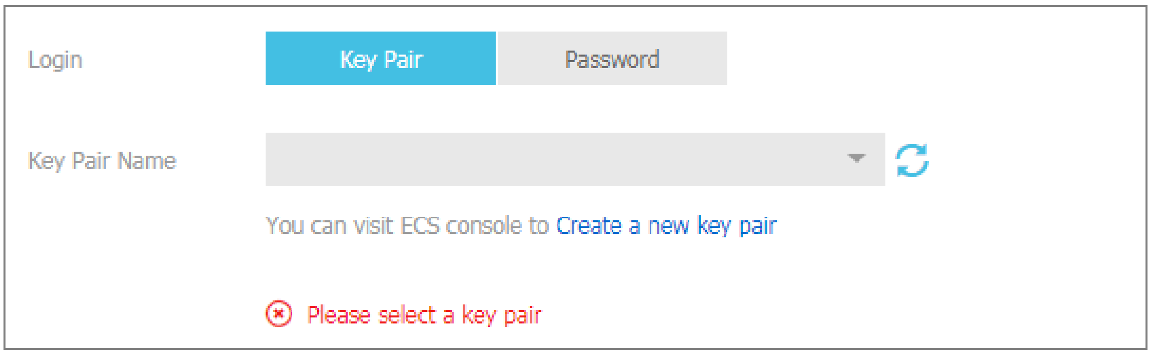
Worker Instance: Create | Add

You can now convert a paid instance to an example of an annual subscription through the ECS Management Console. View details

10. Set the logon mode.

- Set the key pair.

  When creating a cluster, select the key pair logon mode and click **Create a new key pair**. In the ECS console, create a key pair. For details, see *Create an SSH key pair*. After the key pair is created, set the key pair as the credentials for logging on to the cluster.


Login Key Pair Password

Key Pair Name

You can visit ECS console to Create a new key pair

Please select a key pair

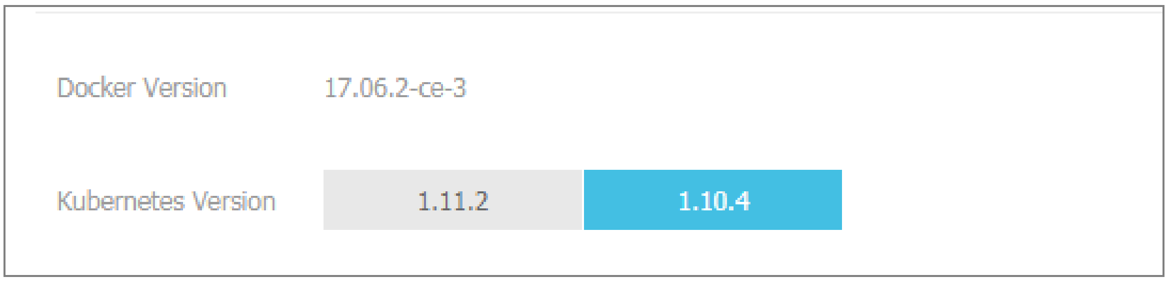
- Set the password.
  - **Logon Password**: Set the node logon password.
  - **Confirm Password**: Confirm your node logon password.

11. Set the**Pod Network CIDR** and **Service CIDR** parameters.

Note:
These two parameters are available only when you select to **Use Existing** VPC.

Specify **Pod Network CIDR** and **Service CIDR**. Both of them cannot overlap with the Classless Inter-Domain Routing (CIDR) block used by VPC and the existing Kubernetes clusters in VPC. The values cannot be modified after the cluster is created. In addition, service CIDR cannot overlap with pod network CIDR. For more information about how to plan Kubernetes CIDR blocks, see *Plan Kubernetes CIDR blocks under VPC*.

12. Available Docker versions and Kubernetes versions are displayed. You can view the versions and select a version according to your needs.

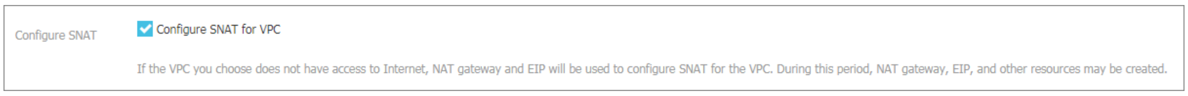

Docker Version 17.06.2-ce-3

Kubernetes Version 1.11.2 1.10.4

13. Select whether to configure a SNAT Gateway for a VPC.

Note:
If you select **Auto Create**, you must configure a SNAT Gateway. If you select **Use Existing**, you can select whether to automatically configure a SNAT Gateway. If you select not to automatically configure a SNAT Gateway, you can configure a NAT Gateway for VPC instances to securely access the Internet, or you can configure a SNAT Gateway manually.

Otherwise, instances in the VPC cannot access the Internet, which causes cluster creation failure.

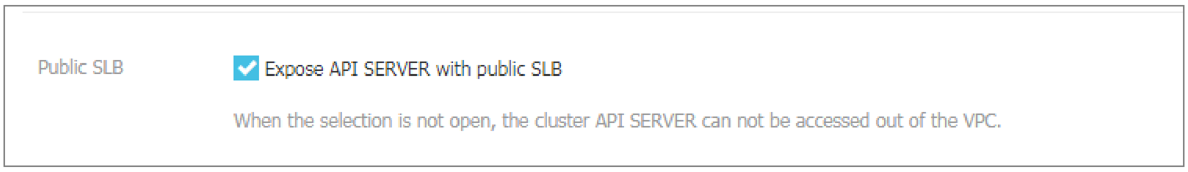

Configure SNAT ✓ Configure SNAT for VPC

If the VPC you choose does not have access to Internet, NAT gateway and EIP will be used to configure SNAT for the VPC. During this period, NAT gateway, EIP, and other resources may be created.

14. Select whether to enable **Use Public SLB to Expose API Server**.

API server provides add, delete, edit, check, watch, and other HTTP Rest interfaces for a variety of resource objects (such as pods and services).

1. If you select to enable this option, the Internet SLB is created and the port 6443 of the Master nodes is exposed. The port corresponds to the API server. Then you can use kubeconfig to connect to and operate the clusters through the Internet.
2. If you select not to enable this option, the Internet SLB is not created. You can only use kubeconfig to connect to and operate the clusters inside the VPC.


Public SLB ✓ Expose API SERVER with public SLB

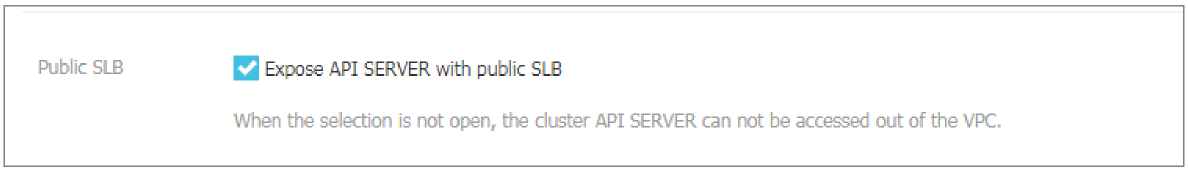
When the selection is not open, the cluster API SERVER can not be accessed out of the VPC.

15. Select whether to enable SSH logon for Internet.

Note:

To enable SSH access for Internet, you must select **Use Public SLB to Expose API Server**.

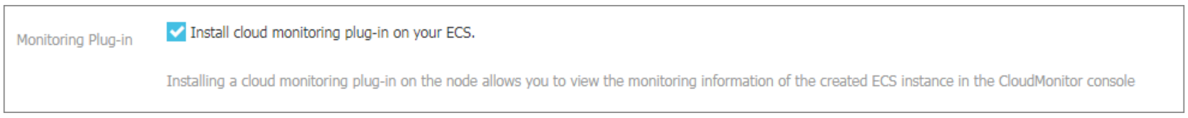
- If you select to enable SSH access for Internet, you can use SSH to access a cluster.
- If you select not to enable SSH access for Internet, you cannot access a cluster by using SSH or connect to a cluster by using kubectl. To access a cluster instance by using SSH, manually bind an EIP to the ECS instance, configure security group rules, and open the SSH port (22). For details, see *Access Kubernetes clusters by using SSH*.


Public SLB ✓ Expose API SERVER with public SLB

When the selection is not open, the cluster API SERVER can not be accessed out of the VPC.

16. Select whether to install a cloud monitoring plug-in on your ECS.

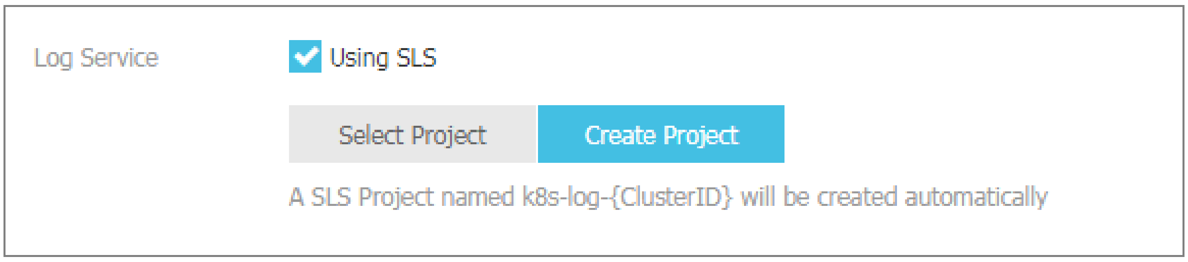
You can install a cloud monitoring plug-in on the ECS node to view the monitoring information of the created ECS instances in the CloudMonitor console.


Monitoring Plug-in ✓ Install cloud monitoring plug-in on your ECS.

Installing a cloud monitoring plug-in on the node allows you to view the monitoring information of the created ECS instance in the CloudMonitor console

17. Select whether to use Log Service. You can select an existing project or create a project.

If you select **Using SLS**, the Log Service plug-in is automatically configured in the cluster. When creating an application, you can quickly use Log Service with a simple configuration. For details, see *Use Log Service to collect Kubernetes cluster logs*.


Log Service ✓ Using SLS

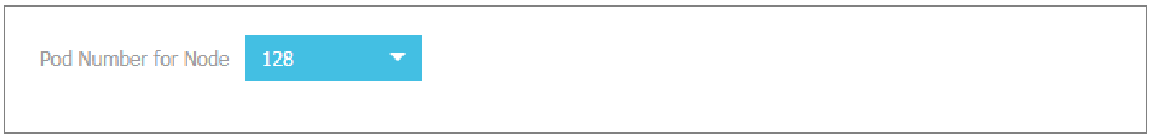
A SLS Project named k8s-log-{ClusterID} will be created automatically

18. Select whether to show advance config.

a. Select a network plug-in. Available network plug-ins are Flannel and Terway. For details, see *Do I select the Terway or Flannel plug-in for my Kubernetes cluster network?*.

- Flannel: a simple and stable community Flannel CNI plug-in. It provides only a few simple features. For example, it does not support the Kubernetes Network Policy.
- Terway: a network plug-in developed by Alibaba Cloud Container service. It 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, it supports bandwidth limiting for individual containers.

b. Set the number of pods for a node, that is, the maximum number of pods that can be run by a single node. We recommend that you use the default value.


Pod Number for Node 128

c. Select whether to use **Custom Cluster CA**. If this option is selected, the CA certificate can be added to the Kubernetes cluster, which enhances the security of information exchange between the server and client.

Cluster CA Custom Cluster CA

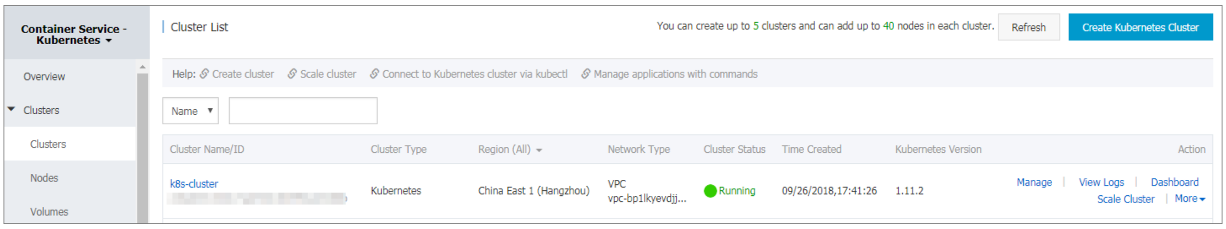
19. Click **Create** in the upper-right corner.

Note:

A multi-node Kubernetes cluster typically takes 10 minutes to be created.

View cluster deployment results.

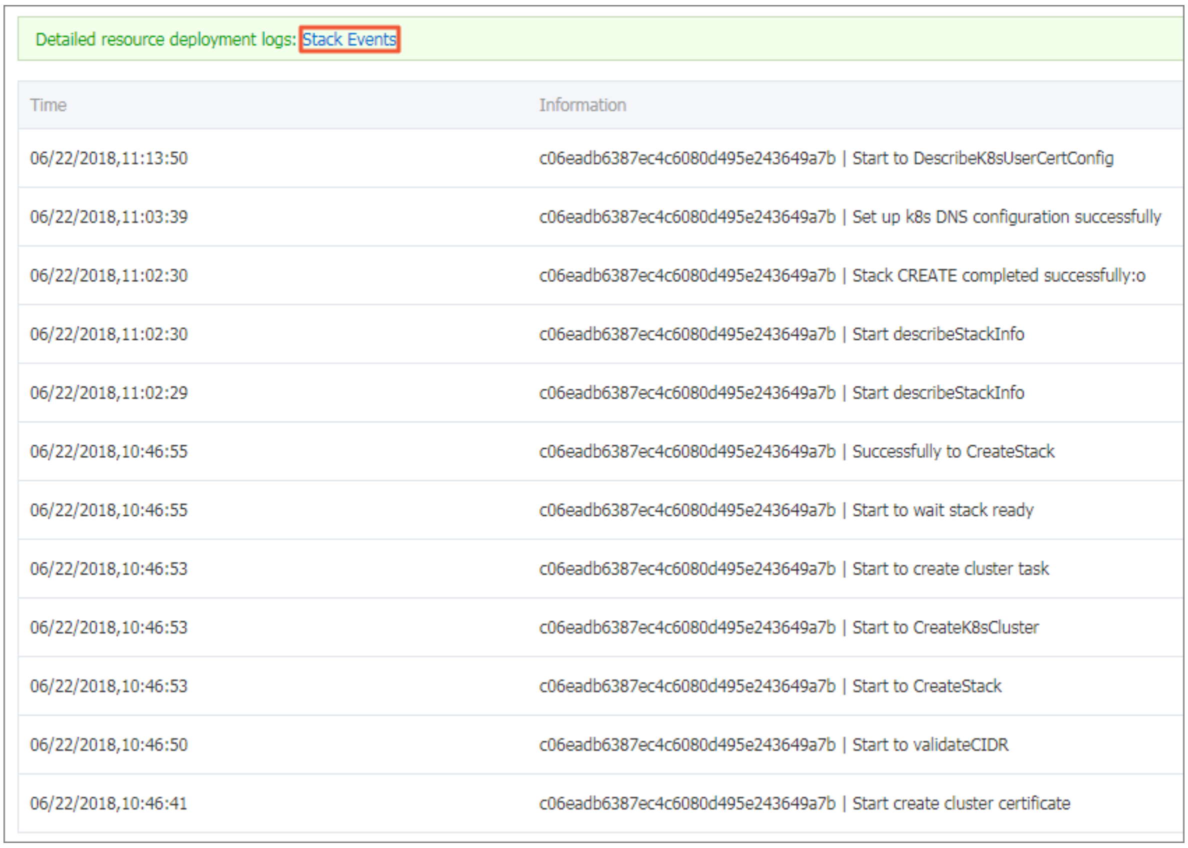
After the cluster is successfully created, you can view the cluster in the Cluster List of the Container Service - Kubernetes console.


Container Service - Kubernetes
Overview
Clusters
Clusters
Nodes
Volumes
Cluster List
Refresh
Create Kubernetes Cluster
Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands
Name

You can create up to 5 clusters and can add up to 40 nodes in each cluster.

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
k8s-cluster [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1lkyevdjj...	Running	09/26/2018,17:41:26	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More

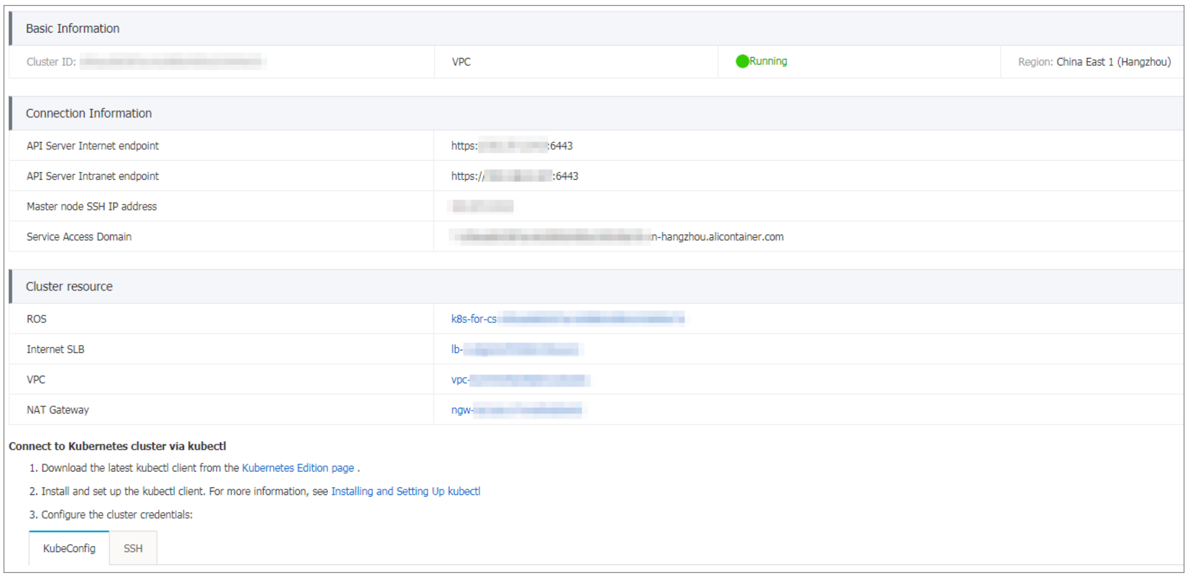
- Click **View Logs** on the right of the cluster to view the cluster logs. To view more detailed information, click **Stack Events**.



Detailed resource deployment logs: Stack Events

Time	Information
06/22/2018,11:13:50	c06eadb6387ec4c6080d495e243649a7b | Start to DescribeK8sUserCertConfig
06/22/2018,11:03:39	c06eadb6387ec4c6080d495e243649a7b | Set up k8s DNS configuration successfully
06/22/2018,11:02:30	c06eadb6387ec4c6080d495e243649a7b | Stack CREATE completed successfully:o
06/22/2018,11:02:30	c06eadb6387ec4c6080d495e243649a7b | Start describeStackInfo
06/22/2018,11:02:29	c06eadb6387ec4c6080d495e243649a7b | Start describeStackInfo
06/22/2018,10:46:55	c06eadb6387ec4c6080d495e243649a7b | Successfully to CreateStack
06/22/2018,10:46:55	c06eadb6387ec4c6080d495e243649a7b | Start to wait stack ready
06/22/2018,10:46:53	c06eadb6387ec4c6080d495e243649a7b | Start to create cluster task
06/22/2018,10:46:53	c06eadb6387ec4c6080d495e243649a7b | Start to CreateK8sCluster
06/22/2018,10:46:53	c06eadb6387ec4c6080d495e243649a7b | Start to CreateStack
06/22/2018,10:46:50	c06eadb6387ec4c6080d495e243649a7b | Start to validateCIDR
06/22/2018,10:46:41	c06eadb6387ec4c6080d495e243649a7b | Start create cluster certificate

- You can also click **Manage** on the right of the cluster to view the basic information and connection information about this cluster.



Basic Information

Cluster ID: [redacted]	VPC	●Running	Region: China East 1 (Hangzhou)

Connection Information

API Server Internet endpoint	https:[redacted]:6443
API Server Intranet endpoint	https:/[redacted]:6443
Master node SSH IP address	[redacted]
Service Access Domain	[redacted]n-hangzhou.alicontainer.com

Cluster resource

ROS	k8s-for-cs[redacted]
Internet SLB	lb-[redacted]
VPC	vpc-[redacted]
NAT Gateway	ngw-[redacted]

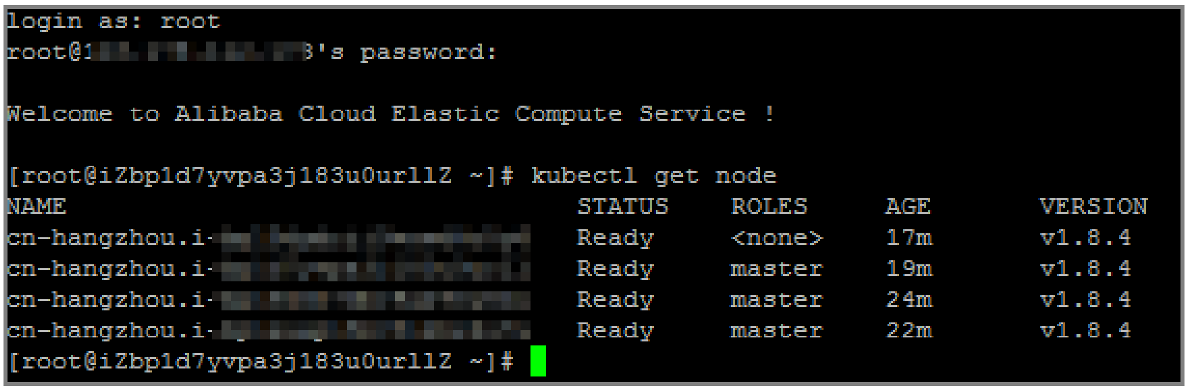
Connect to Kubernetes cluster via kubectl

1. Download the latest kubectl client from the Kubernetes Edition page .
2. Install and set up the kubectl client. For more information, see Installing and Setting Up kubectl
3. Configure the cluster credentials:

In the Cluster Info section:

- **API Server Internet endpoint**: The IP address and port through which the Kubernetes API server provides services for the Internet. It enables you to manage the cluster by using kubectl or other tools on your terminal.
- **API Server Intranet endpoint**: The IP address and port through which the Kubernetes API server provides services inside the cluster. This IP address is the address of the SLB instance, and three Master nodes in the backend provide the services.
- **Master node SSH IP address**: You can directly log on to the Master nodes by using SSH to perform routine maintenance for the cluster.
- **Service Access Domain**: Provides the services in the cluster with access domain name for testing. The service access domain name suffix is `<cluster_id>.<region_id>.alicontainer.com`.

For example, you can log on to the Master nodes by using SSH, and run `kubectl get node` to view the node information of the cluster.



```
login as: root
root@1[redacted]3's password:

Welcome to Alibaba Cloud Elastic Compute Service !

[root@iZbp1d7yvpa3j183u0url1Z ~]# kubectl get node
NAME                                 STATUS    ROLES     AGE       VERSION
cn-hangzhou.i-[redacted]             Ready     <none>    17m       v1.8.4
cn-hangzhou.i-[redacted]             Ready     master    19m       v1.8.4
cn-hangzhou.i-[redacted]             Ready     master    24m       v1.8.4
cn-hangzhou.i-[redacted]             Ready     master    22m       v1.8.4
[root@iZbp1d7yvpa3j183u0url1Z ~]#
```

As shown in the preceding figure, the cluster has four nodes, including three Master nodes and one Worker node configured when we set the parameters.

1.3.3 Create a managed Kubernetes cluster

You can create a managed Kubernetes cluster quickly and easily in the Container Service console.

Prerequisites

You have activated the following services: Container Service, Resource Orchestration Service (ROS), Resource Access Management (RAM), and Auto Scaling service.

Log on to the *Container Service console*, *ROS console*, *RAM console*, and *Auto Scaling console* to activate the corresponding services.

Note:

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

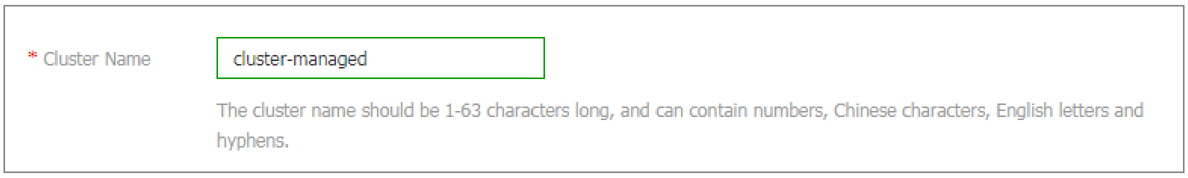
Context

- The SLB instances created with the cluster support only the Pay-As-You-Go billing method.
- Kubernetes clusters support only the Virtual Private Cloud (VPC) network type.
- By default, each account has a specified quota for the cloud resources it can create. If the number of cloud resources exceeds the quota, the account cannot create a cluster. Make sure you have enough quota before creating a cluster. To increase your quota, open a ticket.
  - 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.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane. The **Cluster List** page is displayed. Click **Create Kubernetes Cluster** in the upper-right corner.
3. On the **Create Kubernetes Cluster** page, click **Managed Kubernetes (beta)**.
4. Enter the cluster name.

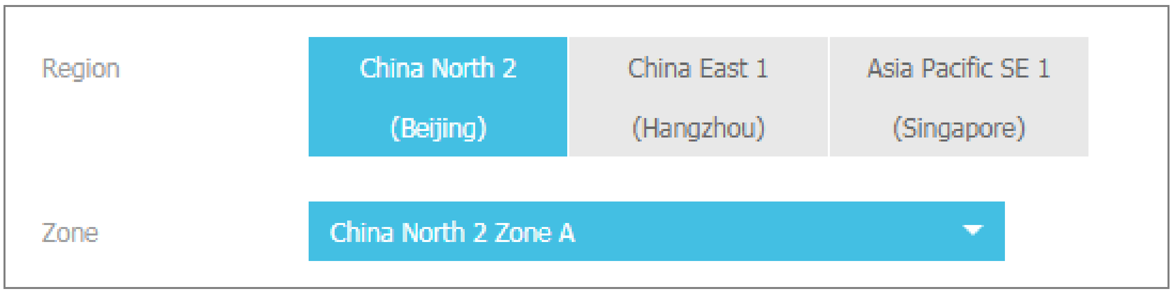
The cluster name can be 1–63 characters long and contain numbers, Chinese characters, English letters, and hyphens (-).



* Cluster Name: cluster-managed

The cluster name should be 1-63 characters long, and can contain numbers, Chinese characters, English letters and hyphens.

5. Select the region and zone where the cluster is located.



Region: China North 2 (Beijing) | China East 1 (Hangzhou) | Asia Pacific SE 1 (Singapore)

Zone: China North 2 Zone A

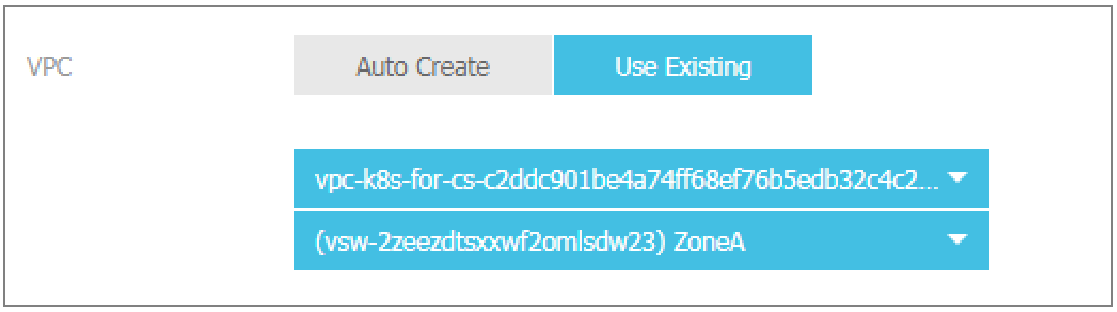
6. Set the cluster network type.

Note:
Kubernetes clusters support only the VPC network type.

VPC: You can select **Auto Create** to create a VPC together with the Kubernetes cluster, or select**Use Existing** to use an existing VPC. If you select **Use Existing**, you can select a VPC and VSwitch from the two displayed drop-down lists.

- **Auto Create**: The system automatically creates a NAT Gateway for your VPC when a cluster is created.
- **Use Existing**: If the selected VPC has a NAT Gateway, Container Service uses the NAT Gateway. Otherwise, the system automatically creates a NAT Gateway by default. If you do not want the system to automatically create a NAT Gateway, deselect the **Configure SNAT for VPC** check box.

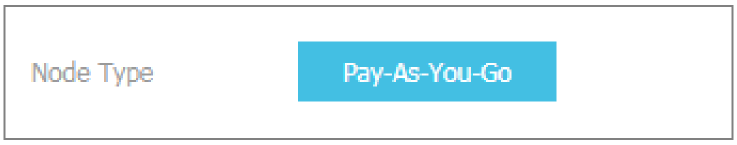
  Note:
  If you deselect the check box, configure the NAT Gateway on your own to implement the VPC Internet environment with secure access, or manually configure the SNAT. Otherwise, instances in the VPC cannot access the Internet normally, which leads to cluster creation failure.


VPC Auto Create Use Existing
vpc-k8s-for-cs-c2ddc901be4a74ff68ef76b5edb32c4c2...
(vsw-2zeezdtsxxwf2omlsdw23) ZoneA

7. Set the node type.

Note:

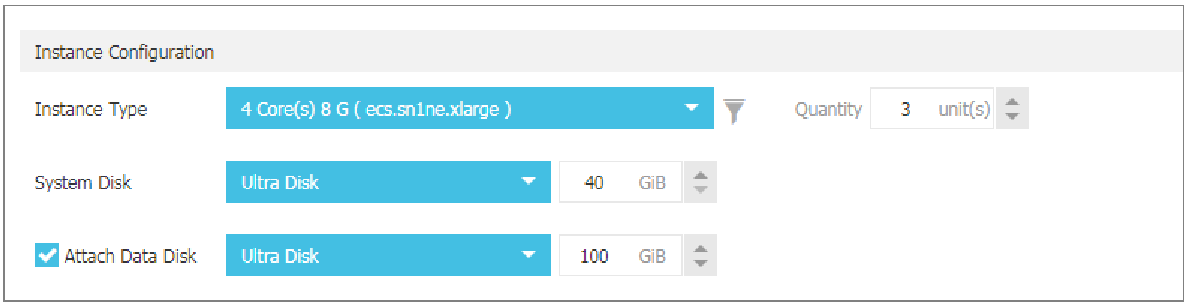
Pay-As-You-Go and **Subscription** types are supported.


Node Type Pay-As-You-Go

8. Configure the instance.

Note:

- Currently, only the CentOS operating system is supported.
- Each cluster contains at least two nodes.
- Each cluster contains up to 48 nodes. To create more nodes, open a ticket.
- System disks are attached to the instances by default. Available system disks are Ultra Disks and SSD Disks.
- You can attach a data disk to the instances. The data disk can be an Ultra Disk or an SSD Disk.

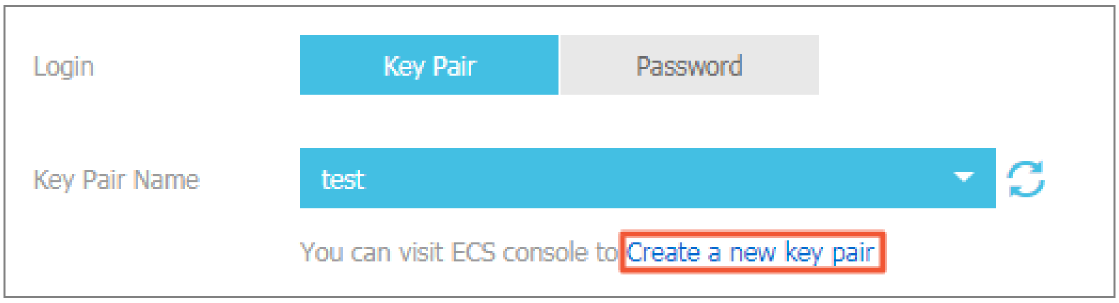

Instance Configuration
Instance Type: 4 Core(s) 8 G (ecs.sn1ne.xlarge) Quantity 3 unit(s)
System Disk: Ultra Disk 40 GiB
✓ Attach Data Disk: Ultra Disk 100 GiB

9. Set the logon mode.

- Set the key pair.

When creating a cluster, select the key pair logon mode and click **Create a new key pair**. In the ECS console, create a key pair. For details, see *Create an SSH key pair*. After the key pair is created, set the key pair as the credentials for logging on to the cluster.

- Set the password.
  - **Logon Password**: Set the node logon password.
  - **Confirm Password**: Confirm your node logon password.



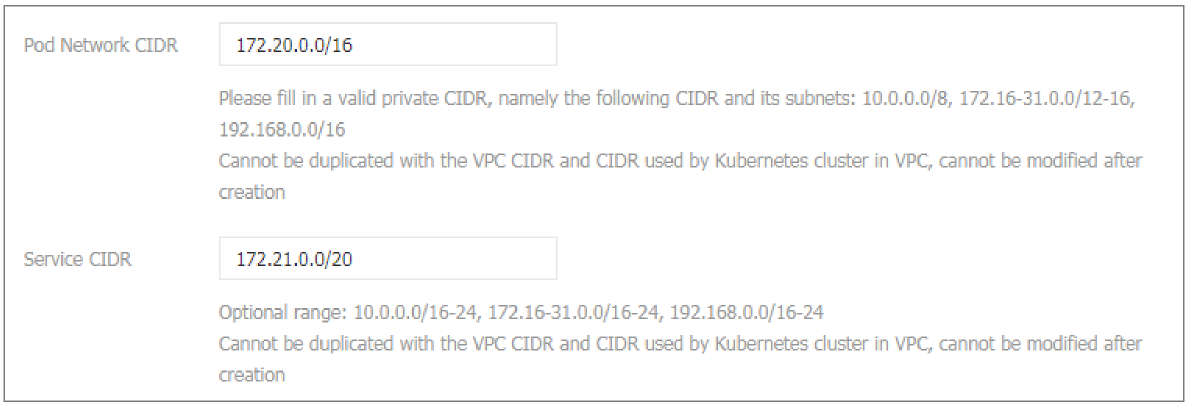
Key Pair Name: test

You can visit ECS console to Create a new key pair

10. Set the **Pod Network CIDR** and **Service CIDR** parameters.

Note:

- These two parameters are available only when you select to **Use Existing** VPC.
- Both **Pod Network CIDR** and **Service CIDR** cannot overlap with the Classless Inter-Domain Routing (CIDR) block used by the VPC and the existing Kubernetes clusters in the VPC. The values cannot be modified after the cluster is created. In addition, service CIDR cannot overlap with pod network CIDR. For more information about how to plan Kubernetes CIDR blocks, see *Plan Kubernetes CIDR blocks under VPC*.



Pod Network CIDR: 172.20.0.0/16

Please fill in a valid private CIDR, namely the following CIDR and its subnets: 10.0.0.0/8, 172.16-31.0.0/12-16, 192.168.0.0/16
Cannot be duplicated with the VPC CIDR and CIDR used by Kubernetes cluster in VPC, cannot be modified after creation

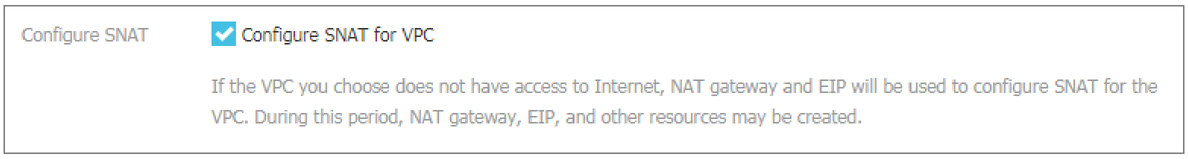
Service CIDR: 172.21.0.0/20

Optional range: 10.0.0.0/16-24, 172.16-31.0.0/16-24, 192.168.0.0/16-24
Cannot be duplicated with the VPC CIDR and CIDR used by Kubernetes cluster in VPC, cannot be modified after creation

11. Select whether to configure a SNAT Gateway for the VPC.

Note:

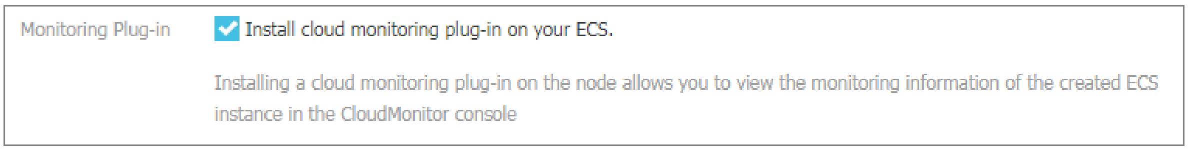
- If you select **Auto Create**, you must configure a SNAT Gateway.
- If you select **Use Existing**, you can select whether to automatically configure a SNAT Gateway. If you select not to automatically configure a SNAT Gateway, you can configure a NAT Gateway for VPC instances to securely access the Internet, or you can configure a SNAT Gateway manually. Otherwise, the instances in the VPC cannot access the Internet, and the cluster fails to be created.


Configure SNAT ✓ Configure SNAT for VPC

If the VPC you choose does not have access to Internet, NAT gateway and EIP will be used to configure SNAT for the VPC. During this period, NAT gateway, EIP, and other resources may be created.

12. Select whether to install a cloud monitoring plug-in on your ECS.

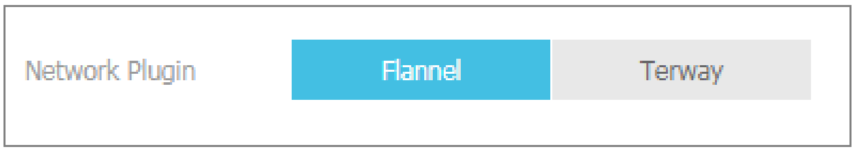
You can install a cloud monitoring plug-in on the ECS node to view the monitoring information of the created ECS instances in the CloudMonitor console.


Monitoring Plug-in ✓ Install cloud monitoring plug-in on your ECS.

Installing a cloud monitoring plug-in on the node allows you to view the monitoring information of the created ECS instance in the CloudMonitor console

13. Select a network plug-in. Available network plug-ins are Flannel and Terway. For details, see *Do I select the Terway or Flannel plug-in for my Kubernetes cluster network?*.

- Flannel: a simple and stable community Flannel plug-in. It provides only a few simple features. For example, it does not support the Kubernetes Network Policy.
- Terway: a network plug-in developed by Alibaba Cloud Container service. It 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, it supports bandwidth limiting for individual containers.

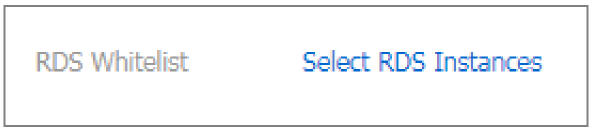


14. Set the RDS whitelist.

Add the IP addresses of the ECS instances to the RDS instance whitelist.

Note:

This option is available only when you select to **Use Existing** VPC.



RDS Whitelist Select RDS Instances

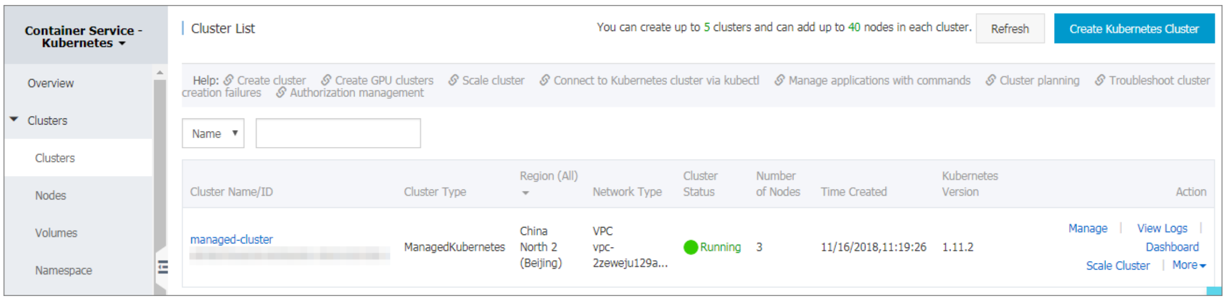
15. Click **Create** in the upper-right corner.

Note:

Normally, it takes five minutes to create a multi-node Kubernetes cluster.

Result

After the cluster is successfully created, you can view the cluster on the **Cluster List** page of the Container Service console.

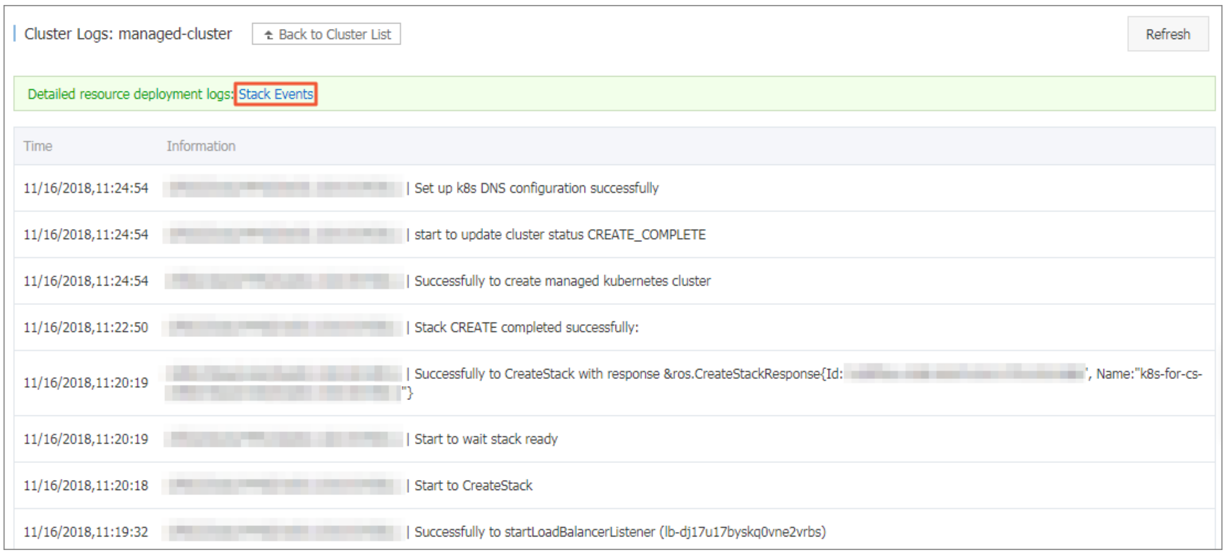

Container Service - Kubernetes
Overview
Clusters
Clusters
Nodes
Volumes
Namespace
Refresh
Create Kubernetes Cluster
Help: Create cluster creation failures
Create GPU clusters
Scale cluster
Connect to Kubernetes cluster via kubectl
Manage applications with commands
Cluster planning
Troubleshoot cluster
Authorization management
Name

Cluster List

You can create up to 5 clusters and can add up to 40 nodes in each cluster.

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Number of Nodes	Time Created	Kubernetes Version	Action
managed-cluster [redacted]	ManagedKubernetes	China North 2 (Beijing)	VPC vpc-2zeweju129a...	● Running	3	11/16/2018,11:19:26	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More

Click **View Logs** on the right of the cluster to view the cluster logs on the **Cluster Logs** page. To view more information, click **Stack Events**.

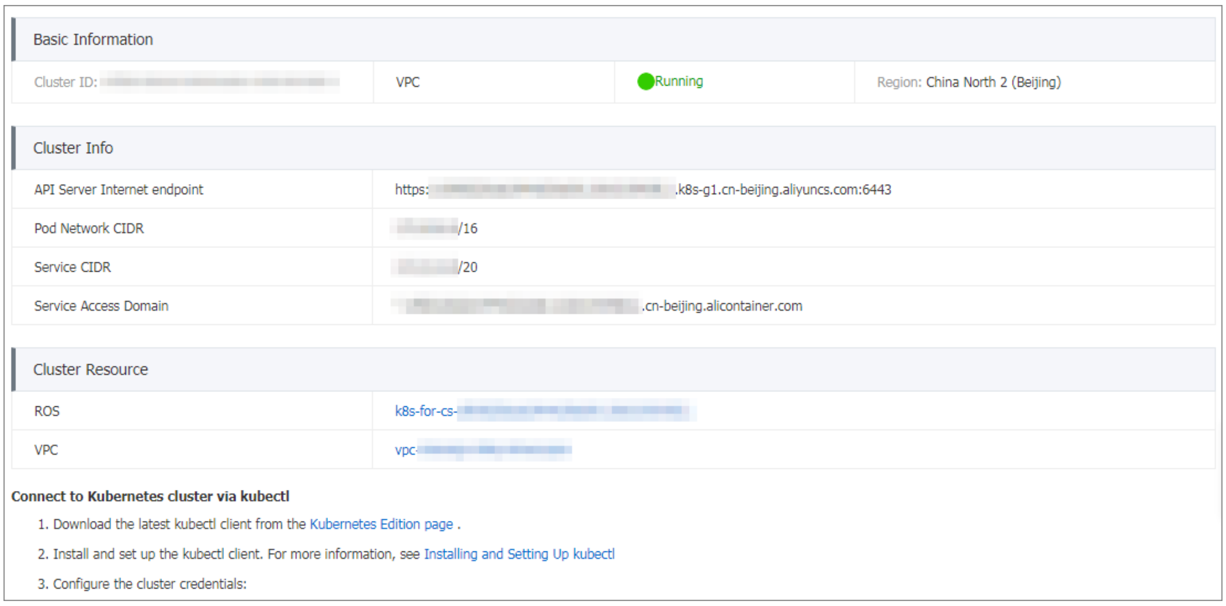


Cluster Logs: managed-cluster

Detailed resource deployment logs: Stack Events

Time	Information
11/16/2018,11:24:54	[redacted] | Set up k8s DNS configuration successfully
11/16/2018,11:24:54	[redacted] | start to update cluster status CREATE_COMPLETE
11/16/2018,11:24:54	[redacted] | Successfully to create managed kubernetes cluster
11/16/2018,11:22:50	[redacted] | Stack CREATE completed successfully:
11/16/2018,11:20:19	[redacted] | Successfully to CreateStack with response &ros.CreateStackResponse{Id: [redacted] ', Name:"k8s-for-cs- [redacted] "}
11/16/2018,11:20:19	[redacted] | Start to wait stack ready
11/16/2018,11:20:18	[redacted] | Start to CreateStack
11/16/2018,11:19:32	[redacted] | Successfully to startLoadBalancerListener (lb-dj17u17byskq0vne2vrbs)

On the **Cluster List** page, find the created cluster and click **Manage** to view the basic information and connection information about this cluster.



Basic Information

Cluster ID: [redacted]	VPC	● Running	Region: China North 2 (Beijing)

Cluster Info

API Server Internet endpoint	https:[redacted].k8s-g1.cn-beijing.aliyuncs.com:6443
Pod Network CIDR	[redacted]/16
Service CIDR	[redacted]/20
Service Access Domain	[redacted].cn-beijing.alicontainer.com

Cluster Resource

ROS	k8s-for-cs-[redacted]
VPC	vpc-[redacted]

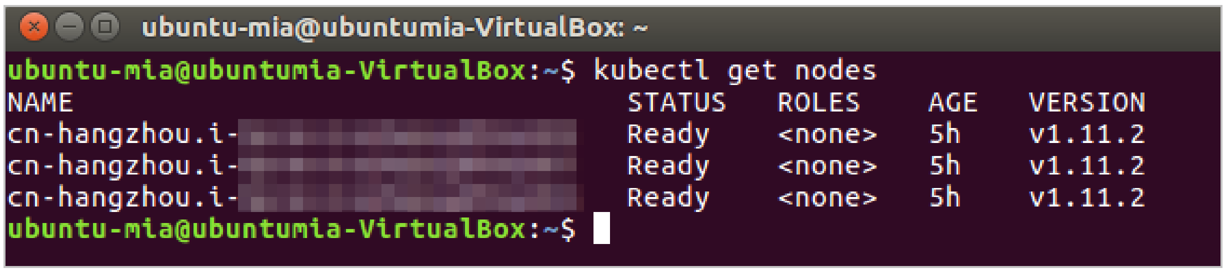
Connect to Kubernetes cluster via kubectl

1. Download the latest kubectl client from the Kubernetes Edition page .
2. Install and set up the kubectl client. For more information, see Installing and Setting Up kubectl
3. Configure the cluster credentials:

In the Cluster Info section:

- **API Server Internet endpoint**: The IP address and port through which the Kubernetes API server provides services for the Internet. With the API Server Internet endpoint, you can manage the cluster by using kubectl or other tools on your terminal.
- **Service Access Domain**: Provides the services in the cluster with access domain name for testing. The service access domain name suffix is `<cluster_id>.<region_id>.alicontainer.com`.

You can see *Connect to a Kubernetes cluster by using kubectl* and run `kubectl get node` to view the node information of the cluster.



```
ubuntu-mia@ubuntumia-VirtualBox: ~
ubuntu-mia@ubuntumia-VirtualBox:~$ kubectl get nodes
NAME                                        STATUS   ROLES    AGE   VERSION
cn-hangzhou.i-[redacted]                    Ready    <none>   5h    v1.11.2
cn-hangzhou.i-[redacted]                    Ready    <none>   5h    v1.11.2
cn-hangzhou.i-[redacted]                    Ready    <none>   5h    v1.11.2
ubuntu-mia@ubuntumia-VirtualBox:~$
```

1.3.4 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.

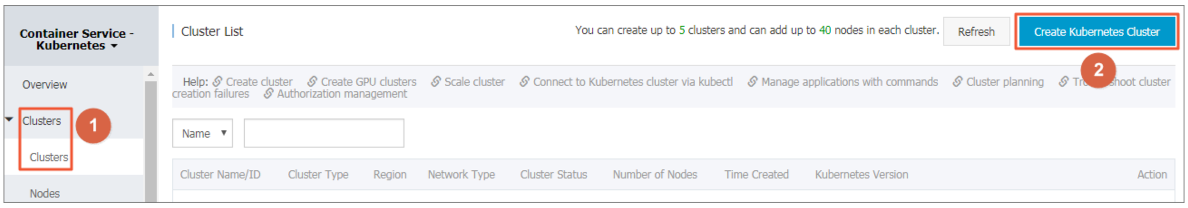
Limits

- The SLB instance created with the 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

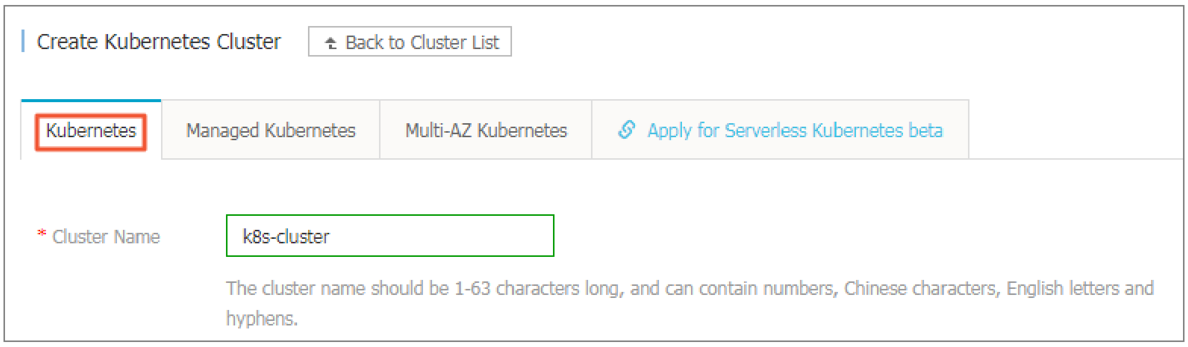
1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, click **Clusters**.
3. Click **Create Kubernetes Cluster** in the upper-right corner.



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*.



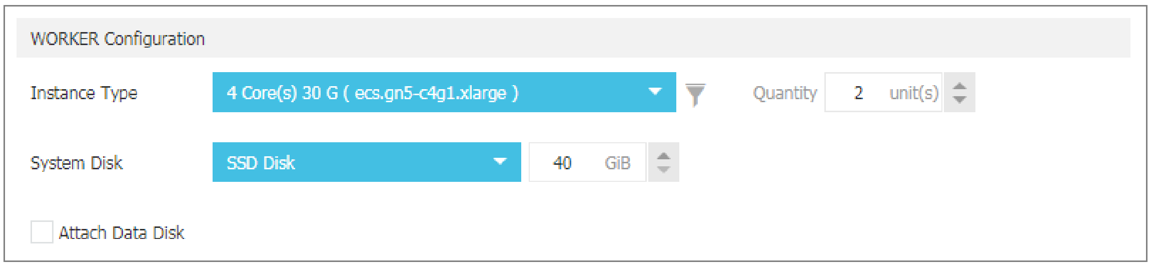
Create Kubernetes Cluster | Back to Cluster List

Kubernetes | Managed Kubernetes | Multi-AZ Kubernetes | Apply for Serverless Kubernetes beta

* Cluster Name: k8s-cluster

The cluster name should be 1-63 characters long, and can contain numbers, Chinese characters, English letters and hyphens.

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.



WORKER Configuration

Instance Type: 4 Core(s) 30 G (ecs.gn5-c4g1.xlarge) Quantity: 2 unit(s)

System Disk: SSD Disk 40 GiB

Attach Data Disk

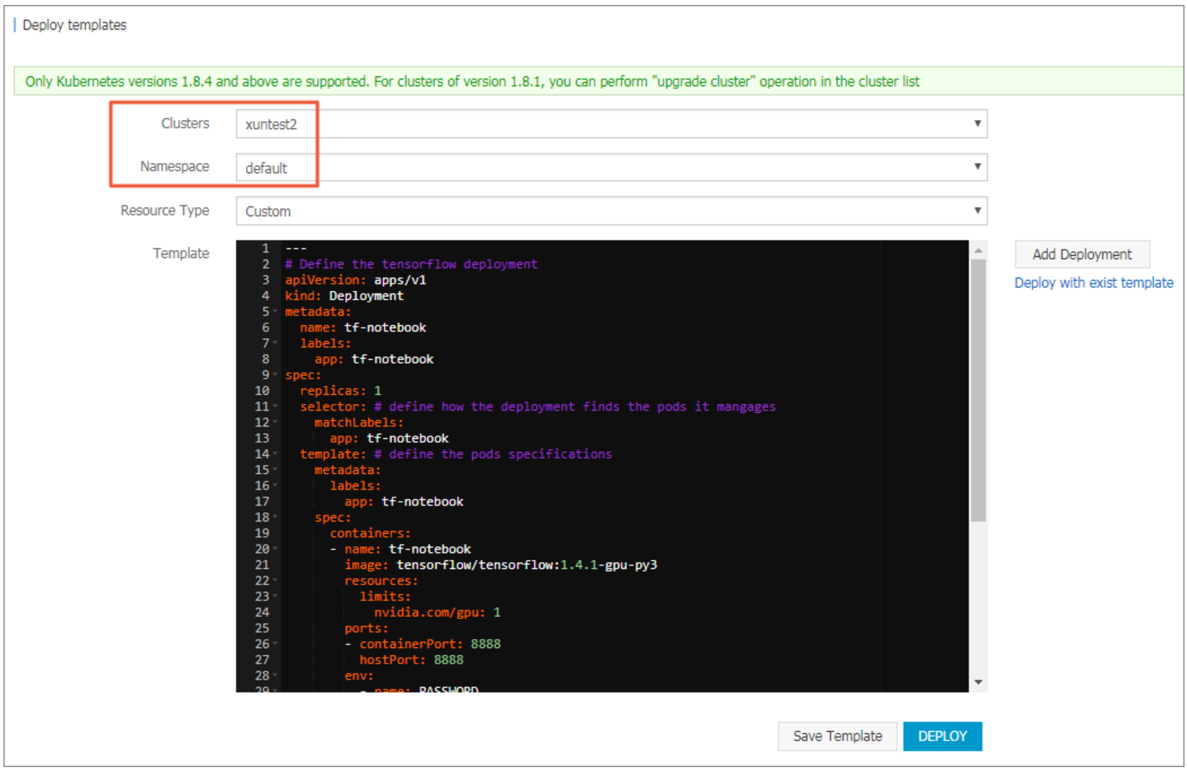
   b. If you choose to add existing instances, you need to have already created GPU cloud servers in the same region where the cluster is to be created.
5. After you have completed all required settings, click **Create** to start cluster deployment.
6. After the cluster is created, choose **Clusters** > **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 **Application** > **Deployment**.

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**.



Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

Clusters: xuntest2
Namespace: default
Resource Type: Custom
Template:

```
---
# Define the tensorflow deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tf-notebook
  labels:
    app: tf-notebook
spec:
  replicas: 1
  selector: # define how the deployment finds the pods it mangages
    matchLabels:
      app: tf-notebook
  template: # define the pods specifications
    metadata:
      labels:
        app: tf-notebook
    spec:
      containers:
      - name: tf-notebook
        image: tensorflow/tensorflow:1.4.1-gpu-py3
        resources:
          limits:
            nvidia.com/gpu: 1
        ports:
        - containerPort: 8888
          hostPort: 8888
        env:
          - name: PASSWORD
```

In this example, a Jupyter application template is orchestrated. The template includes a deployment and a service.

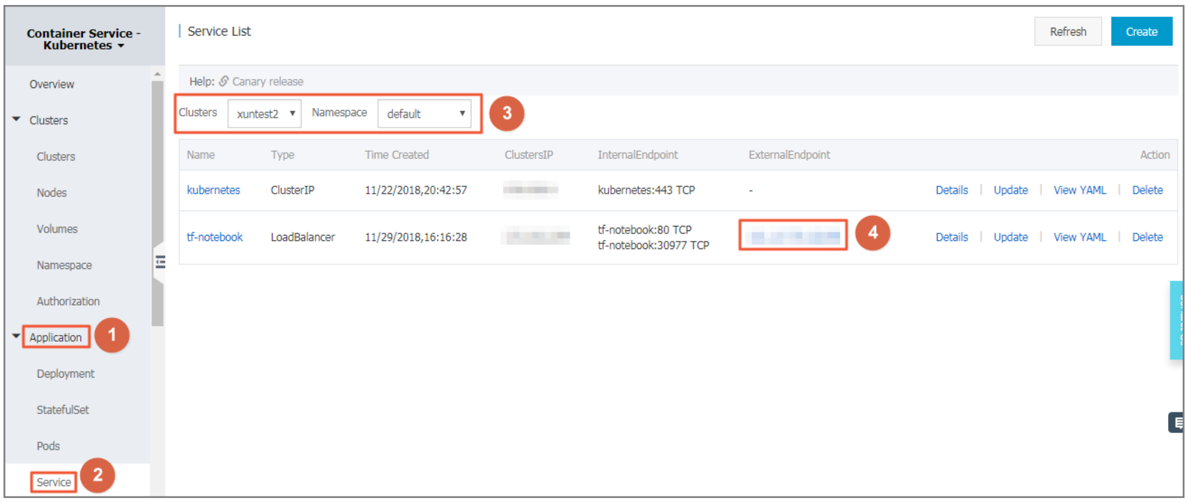
```
---
# Define the tensorflow deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tf-notebook
  labels:
    app: tf-notebook
spec:
  replicas: 1
  selector: # define how the deployment finds the pods it manages
    matchLabels:
      app: tf-notebook
  template: # define the pods specifications
    metadata:
      labels:
        app: tf-notebook
    spec:
      containers:
      - name: tf-notebook
        image: tensorflow/tensorflow:1.4.1-gpu-py3
        resources:
          limits:
            nvidia.com/gpu: 1                      #specify the
number of NVIDIA GPUs that are called by the application
        ports:
        - containerPort: 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: LoadBalancer                           #set Alibaba Cloud
SLB service for the application 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.

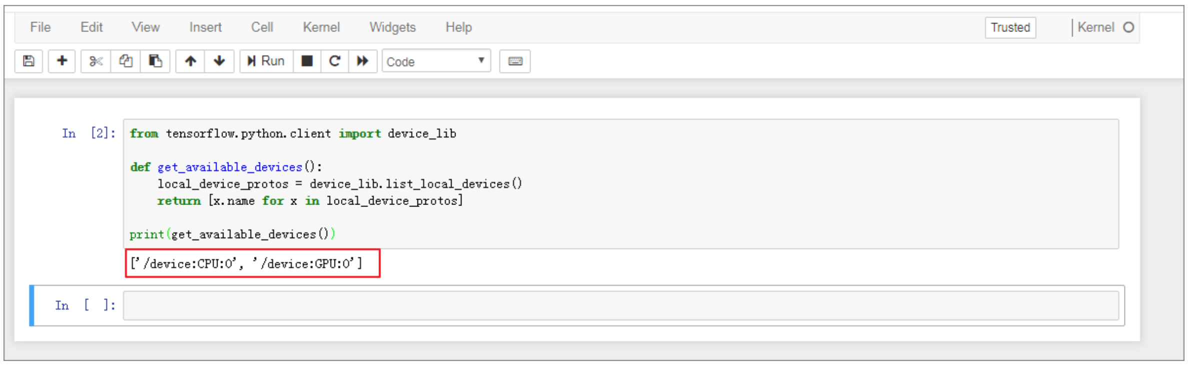
5. In the left-side navigation pane, choose **Application** > **Service**, select the target cluster and namespace, and then view the external endpoint of the tf-notebook service.



Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
kubernetes	ClusterIP	11/22/2018,20:42:57	[redacted]	kubernetes:443 TCP	-	Details | Update | View YAML | Delete
tf-notebook	LoadBalancer	11/29/2018,16:16:28	[redacted]	tf-notebook:80 TCP tf-notebook:30977 TCP	[redacted]	Details | Update | View YAML | Delete

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_available_devices():
    local_device_protos = device_lib.list_local_devices()
    return [x.name for x in local_device_protos]

print(get_available_devices())
```



```
In [2]: from tensorflow.python.client import device_lib

        def get_available_devices():
            local_device_protos = device_lib.list_local_devices()
            return [x.name for x in local_device_protos]

        print(get_available_devices())

        ['/device:CPU:0', '/device:GPU:0']

In [ ]:
```

1.3.5 Create a multi-zone Kubernetes cluster

You can create a multi-zone Kubernetes cluster to guarantee high availability.

Prerequisites

- You have activated the following services: Container Service, Resource Orchestration Service (ROS), Resource Access Management (RAM), and Auto Scaling service.

Log on to the *Container Service console*, *ROS console*, *RAM console*, and *Auto Scaling 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, activate ROS before creating a Kubernetes cluster.

- You must create a Virtual Private Cloud (VPC) and create at least three VSwitches in the VPC . To achieve high availability, we recommend that you create VSwitches in different availability zones.
- You need to manually configure SNAT for each VSwitch in the VPC. Otherwise, instances in the VPC cannot access the Internet normally.

Context

You can create Kubernetes clusters with ECS instances in different availability zones by using the Container Service console to achieve high availability.

Context

During cluster creation, Container Service performs the following operations:

- Create Elastic Compute Service (ECS) instances and configure to log on to other nodes from management nodes with the SSH public key. Install and configure the Kubernetes cluster by using CloudInit.
- Create a security group. This security group allows the VPC inbound access of all the ICMP ports.
- Create a RAM user and an AccessKey. The RAM user has the permissions for querying, creating, and deleting ECS instances, the permissions for adding and deleting cloud disks, and all permissions for the operations on Server Load Balancer (SLB), CloudMonitor, VPC, Log Service, and Network Attached Storage (NAS). The Kubernetes cluster dynamically creates SLB instances, cloud disks, and VPC routing rules according to your configurations.
- Create an intranet SLB instance and expose the port 6443.
- Create an Internet SLB instance and expose the port 6443. (If you enable the SSH logon for Internet when creating the cluster, port 22 is exposed. Otherwise, port 22 is not exposed.)

Limits

- The SLB instances created with the cluster support only the Pay-As-You-Go billing method.

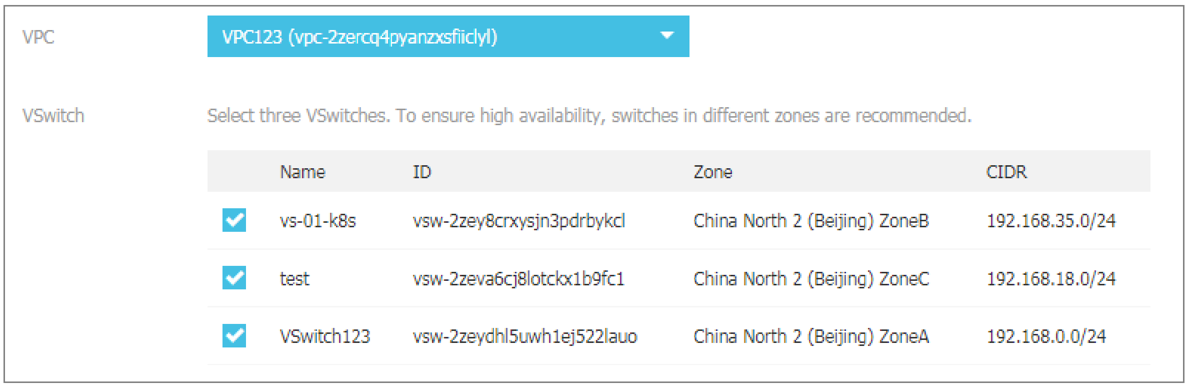
- Kubernetes clusters support only the Virtual Private Cloud (VPC) network type.
- By default, each account has a specified quota for the cloud resources it can create. If the number of cloud resources exceeds the quota, the account cannot create a cluster. Make sure you have enough quota before creating a cluster. To increase your quota, open a ticket.
  - By default, each account can create up to 5 clusters in all regions and add up to 40 nodes to each cluster. To create more clusters or nodes, open a ticket.
  - 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.
- The limits for ECS instances are as follows:
  - Only the CentOS operating system is supported.
  - The Pay-As-You-Go and Subscription ECS instances can be created.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane. Click **Create Kubernetes Cluster** in the upper-right corner.
3. On the Create Kubernetes Cluster page, click **Multi-AZ Kubernetes**.
4. Enter the cluster name.

   The cluster name can be 1–63 characters long and contain numbers, Chinese characters, English letters, and hyphens (-).
5. Select the region where the cluster is located.
6. Select a VPC.

   Select a VPC from the existing VPC drop-down list and select three VSwitches under the VPC. To achieve high availability, we recommend that you select the VSwitches located in different zones.



VPC: VPC123 (vpc-2zercq4pyanzxsfiiclyl)

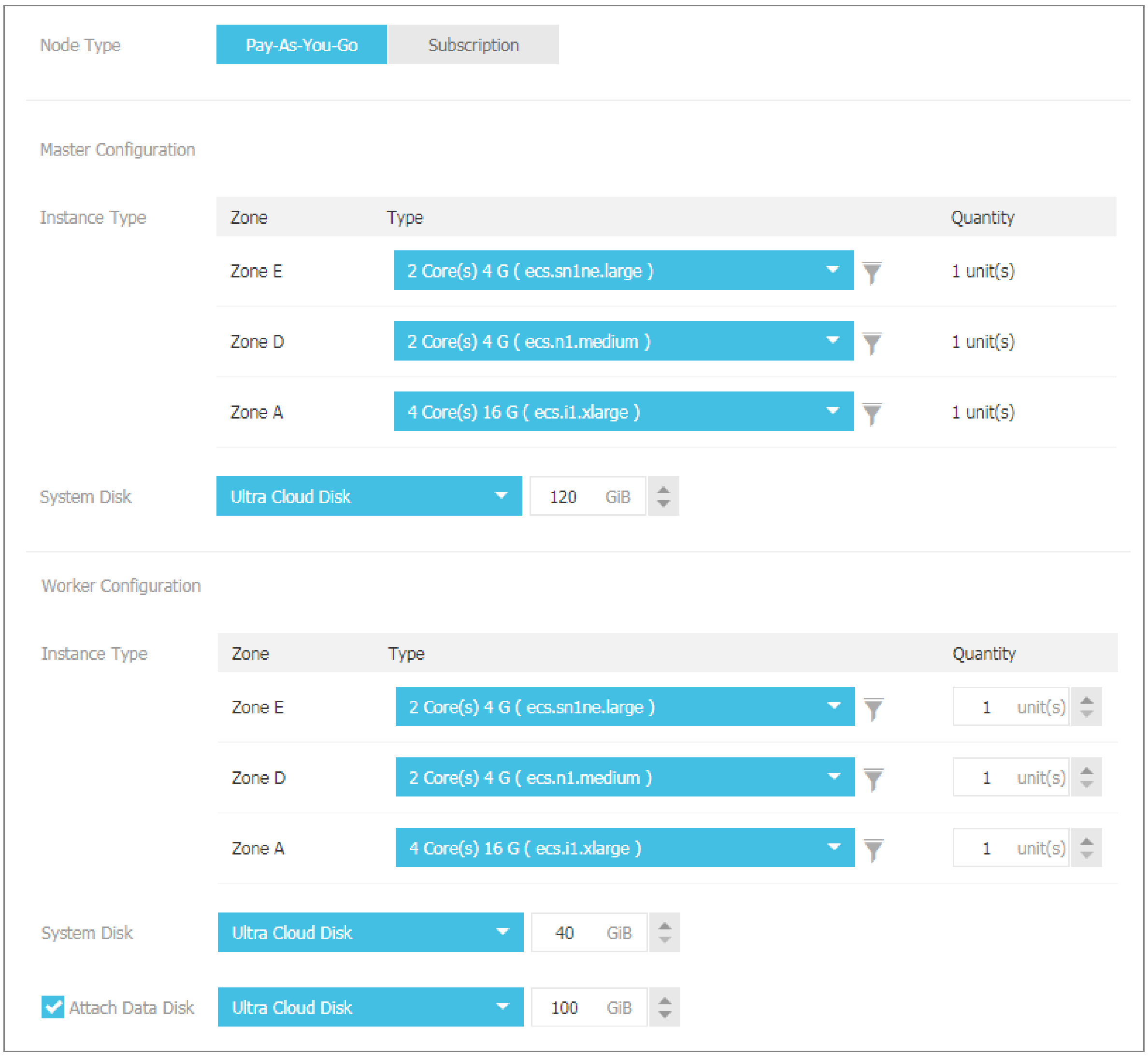
VSwitch: Select three VSwitches. To ensure high availability, switches in different zones are recommended.

	Name	ID	Zone	CIDR
✓	vs-01-k8s	vsw-2zey8crxysjn3pdrbykcl	China North 2 (Beijing) ZoneB	192.168.35.0/24
✓	test	vsw-2zeva6cj8lotckx1b9fc1	China North 2 (Beijing) ZoneC	192.168.18.0/24
✓	VSwitch123	vsw-2zeydhl5uwh1ej522lauo	China North 2 (Beijing) ZoneA	192.168.0.0/24

7. Configure the Master nodes and Worker nodes.
    a) Select a node payment type from Pay-As-You-Go and Subscription.
    b) Select instance types of the Master nodes and Worker nodes, and set the number of Worker nodes.

Note:

- Currently, only the CentOS operating system is supported.
- Currently, only three Master nodes can be created.
- Each cluster can contain up to 37 Worker nodes. To create more nodes, open a ticket.
- System disks are attached to Master nodes and Worker nodes by default. Available system disks include SSD Cloud Disks and Ultra Cloud Disks.
- You can also manually attach a data disk to the Worker node. The data disk can be an Ultra Cloud Disk or an SSD Cloud Disk.



Node Type: Pay-As-You-Go | Subscription

Master Configuration

Instance Type

Zone	Type	Quantity
Zone E	2 Core(s) 4 G (ecs.sn1ne.large)	1 unit(s)
Zone D	2 Core(s) 4 G (ecs.n1.medium)	1 unit(s)
Zone A	4 Core(s) 16 G (ecs.i1.xlarge)	1 unit(s)

System Disk: Ultra Cloud Disk 120 GiB

Worker Configuration

Instance Type

Zone	Type	Quantity
Zone E	2 Core(s) 4 G (ecs.sn1ne.large)	1 unit(s)
Zone D	2 Core(s) 4 G (ecs.n1.medium)	1 unit(s)
Zone A	4 Core(s) 16 G (ecs.i1.xlarge)	1 unit(s)

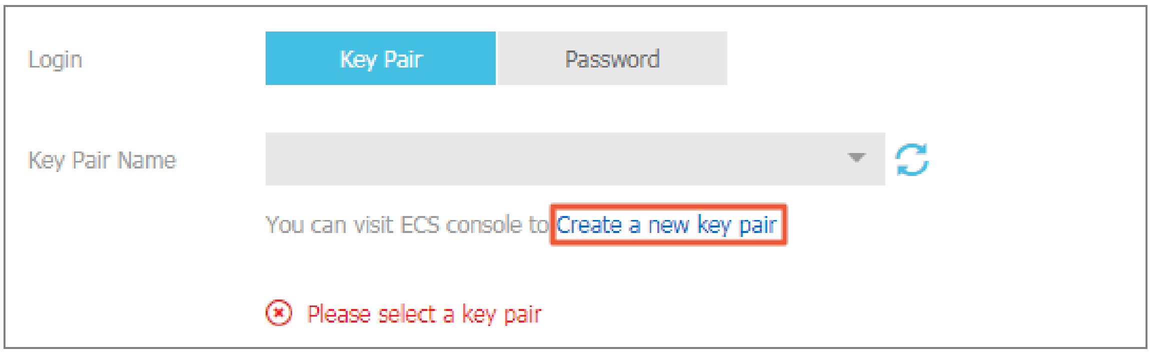
System Disk: Ultra Cloud Disk 40 GiB

✓ Attach Data Disk: Ultra Cloud Disk 100 GiB

8. Configure the logon mode.

   - Set the key pair.

     When creating a cluster, select the key pair logon mode and click **Create a new key pair**. In the ECS console, create a key pair. For details, see *Create an SSH key pair*. After the key pair is created, set the key pair as the credentials for logging on to the cluster.

   - Set the password.

     — **Logon Password**: Set the node logon password.

     — **Confirm Password**: Confirm your node logon password.

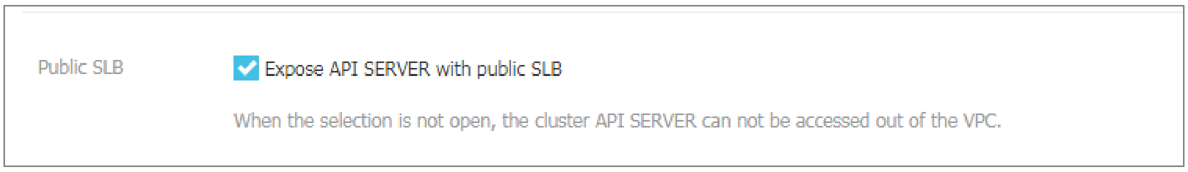
9. Specify the **Pod Network CIDR** and **Service CIDR** parameters.

   Both of them cannot overlap with the Classless Inter-Domain Routing (CIDR) block used by VPC and the existing Kubernetes clusters in VPC. The values cannot be modified after the cluster is created. In addition, service CIDR cannot overlap with pod network CIDR. For more information about how to plan Kubernetes CIDR blocks, see *Plan Kubernetes CIDR blocks under VPC*.

10. Select whether to enable **Use Public SLB to Expose API Server**.

    API server provides add, delete, edit, check, watch, and other HTTP Rest interfaces for a variety of resource objects (such as pods and services).

    1. If you select to enable this option, the Internet SLB is created and the port 6443 of the Master nodes is exposed. The port corresponds to the API server. Then you can use kubeconfig to connect to and operate the clusters through the Internet.
    2. If you select not to enable this option, the Internet SLB is not created. You can only use kubeconfig to connect to and operate the clusters inside the VPC.


Public SLB ✓ Expose API SERVER with public SLB

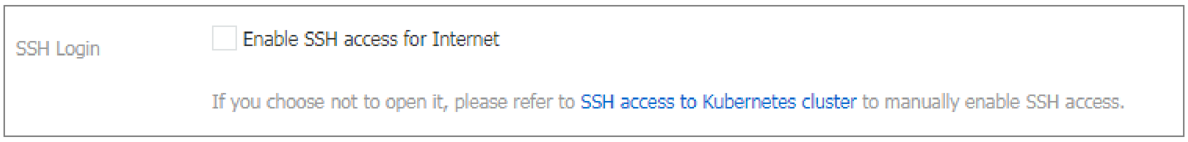
When the selection is not open, the cluster API SERVER can not be accessed out of the VPC.

11. Select whether to enable SSH logon for Internet.

Note:

To enable SSH access for Internet, you must select **Use Public SLB to Expose API Server**.

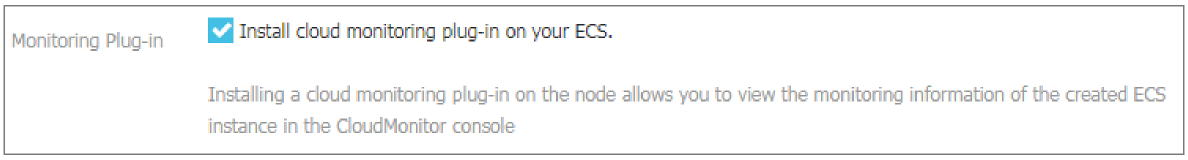
- If you select to enable SSH access for Internet, you can use SSH to access a cluster.
- If you select not to enable SSH access for Internet, you cannot access a cluster by using SSH or connect to a cluster by using kubectl. To access a cluster instance by using SSH, manually bind an EIP to the ECS instance, configure security group rules, and open the SSH port (22). For details, see *Access Kubernetes clusters by using SSH*.


SSH Login ☐ Enable SSH access for Internet

If you choose not to open it, please refer to SSH access to Kubernetes cluster to manually enable SSH access.

12. Select whether to install a cloud monitoring plug-in on your ECS.

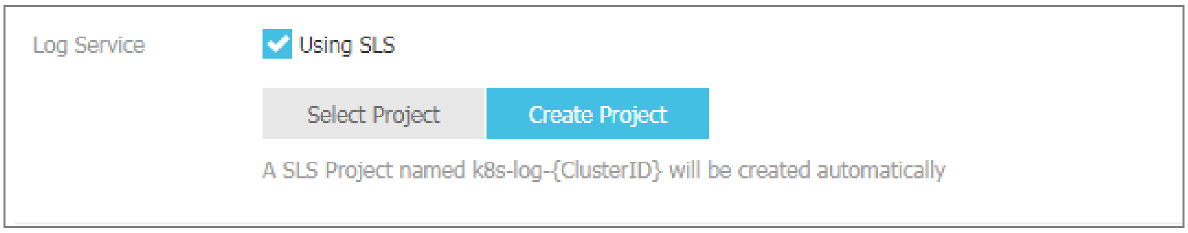
You can install a cloud monitoring plug-in on the ECS node to view the monitoring information of the created ECS instances in the CloudMonitor console.


Monitoring Plug-in ✓ Install cloud monitoring plug-in on your ECS.

Installing a cloud monitoring plug-in on the node allows you to view the monitoring information of the created ECS instance in the CloudMonitor console

13. Select whether to use Log Service. You can select an existing project or create a project.

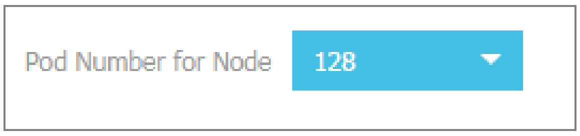
If you select **Using SLS**, the Log Service plug-in is automatically configured in the cluster. When creating an application, you can quickly use Log Service with a simple configuration. For details, see *Use Log Service to collect Kubernetes cluster logs*.


Log Service ✓ Using SLS

A SLS Project named k8s-log-{ClusterID} will be created automatically

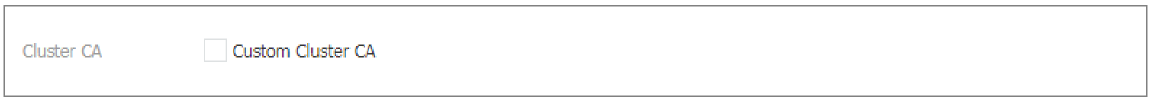
14. Select whether to show advance config.

1. Select a network plug-in. Available network plug-ins are Flannel and Terway. For details, see *Do I select the Terway or Flannel plug-in for my Kubernetes cluster network?*.
   - Flannel: a simple and stable community Flannel CNI plug-in. It provides only a few simple features. For example, it does not support the Kubernetes Network Policy.
   - Terway: a network plug-in developed by Alibaba Cloud Container service. It 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, it supports bandwidth limiting for individual containers.
2. Set the number of pods for a node, that is, the maximum number of pods that can be run by a single node.

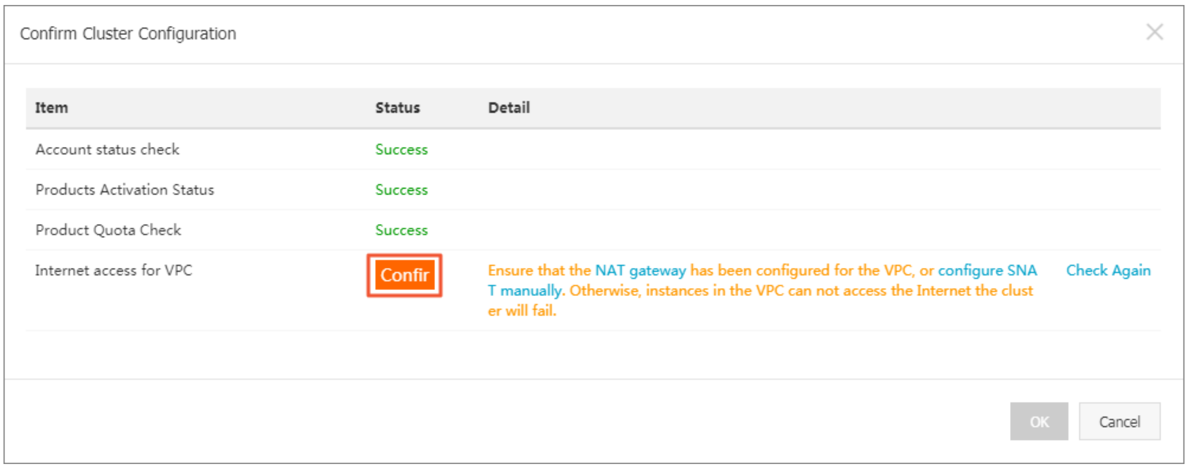

Pod Number for Node 128

3. Select whether to use **Custom Image**. The ECS instance installs the default CentOS version if no custom image is selected.

   Currently, you can only select an image based on CentOS custom version to quickly deploy the environment you need.
4. Sets whether to use **Custom Cluster CA**. If this option is selected, the CA certificate can be added to the Kubernetes cluster, which enhances the security of information exchange between the server and client.


Cluster CA ☐ Custom Cluster CA

15. Click **Create**, confirm the Internet access for VPC in the displayed dialog box, and click **OK** to start the deployment.



Confirm Cluster Configuration

Item	Status	Detail	
Account status check	Success		
Products Activation Status	Success		
Product Quota Check	Success		
Internet access for VPC	Confir	Ensure that the NAT gateway has been configured for the VPC, or configure SNAT manually. Otherwise, instances in the VPC can not access the Internet the cluster will fail.	Check Again

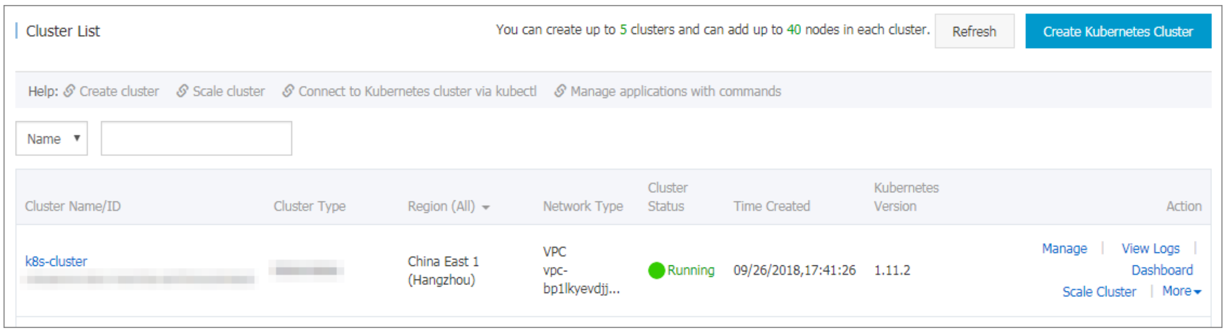
Note:

A multi-node Kubernetes cluster typically takes 10 minutes to be created.

Result

View cluster deployment results.

After the cluster is successfully created, you can view the cluster in the Cluster List of the Container Service console.



Cluster List

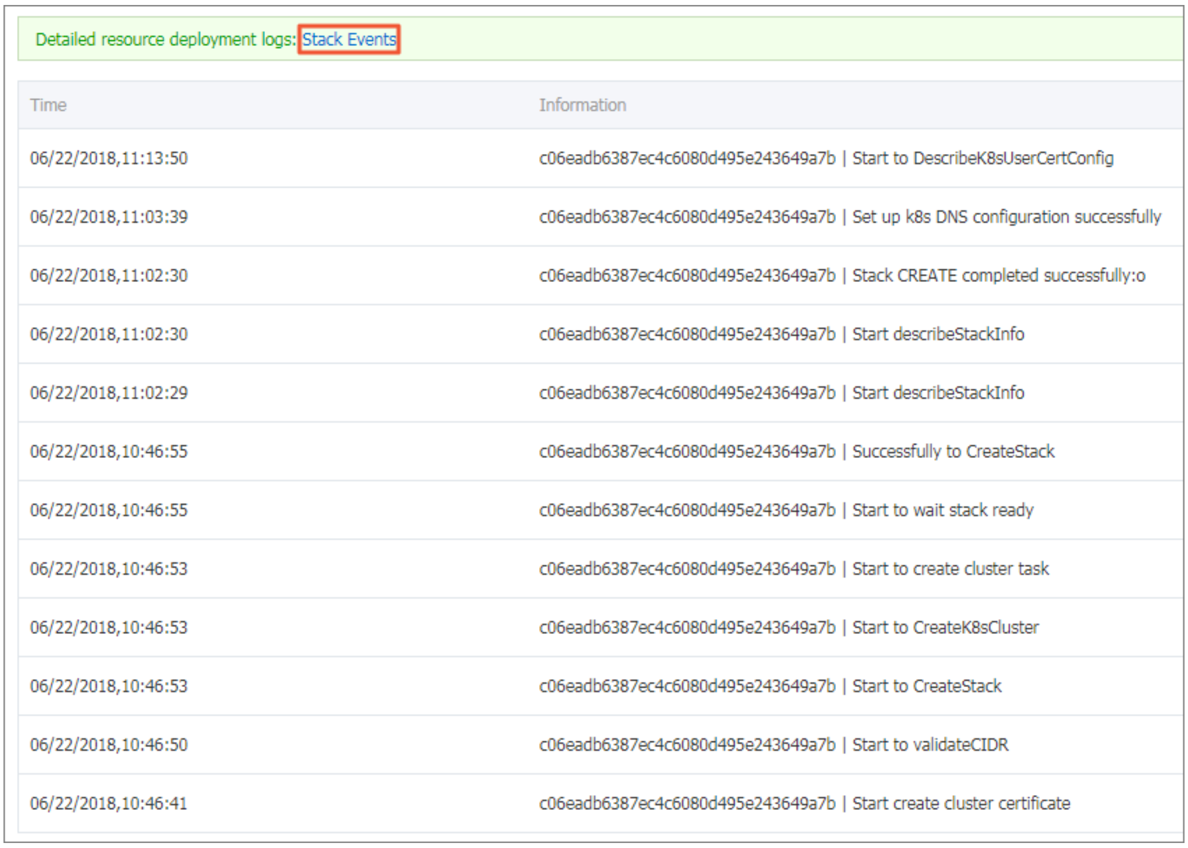
You can create up to 5 clusters and can add up to 40 nodes in each cluster.

Refresh
Create Kubernetes Cluster
Help: Create cluster
Scale cluster
Connect to Kubernetes cluster via kubectl
Manage applications with commands
Name

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
k8s-cluster [redacted]	[redacted]	China East 1 (Hangzhou)	VPC vpc-bp1lkyevdjj...	● Running	09/26/2018,17:41:26	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More

What's next

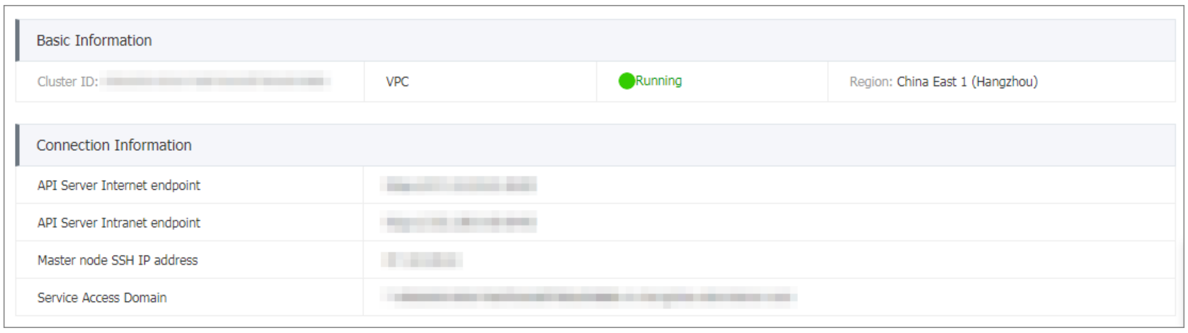
- Click **View Logs** at the right of the cluster to view the cluster logs. To view more detailed information, click **Stack Events**.



Detailed resource deployment logs: Stack Events

Time	Information
06/22/2018,11:13:50	c06eadb6387ec4c6080d495e243649a7b | Start to DescribeK8sUserCertConfig
06/22/2018,11:03:39	c06eadb6387ec4c6080d495e243649a7b | Set up k8s DNS configuration successfully
06/22/2018,11:02:30	c06eadb6387ec4c6080d495e243649a7b | Stack CREATE completed successfully:o
06/22/2018,11:02:30	c06eadb6387ec4c6080d495e243649a7b | Start describeStackInfo
06/22/2018,11:02:29	c06eadb6387ec4c6080d495e243649a7b | Start describeStackInfo
06/22/2018,10:46:55	c06eadb6387ec4c6080d495e243649a7b | Successfully to CreateStack
06/22/2018,10:46:55	c06eadb6387ec4c6080d495e243649a7b | Start to wait stack ready
06/22/2018,10:46:53	c06eadb6387ec4c6080d495e243649a7b | Start to create cluster task
06/22/2018,10:46:53	c06eadb6387ec4c6080d495e243649a7b | Start to CreateK8sCluster
06/22/2018,10:46:53	c06eadb6387ec4c6080d495e243649a7b | Start to CreateStack
06/22/2018,10:46:50	c06eadb6387ec4c6080d495e243649a7b | Start to validateCIDR
06/22/2018,10:46:41	c06eadb6387ec4c6080d495e243649a7b | Start create cluster certificate

- You can also click **Manage** on the right of the cluster to view the basic information and connection information about this cluster.



Basic Information

Cluster ID: [redacted]	VPC	● Running	Region: China East 1 (Hangzhou)

Connection Information

API Server Internet endpoint	[redacted]
API Server Intranet endpoint	[redacted]
Master node SSH IP address	[redacted]
Service Access Domain	[redacted]

In the Cluster Info section:

- **API Server Internet endpoint:** The IP address and port through which the Kubernetes API server provides services for the Internet. It enables you to manage the cluster by using kubectl or other tools on your terminal.
- **API Server Intranet endpoint:** The IP address and port through which the Kubernetes API server provides services inside the cluster. This IP address is the address of the SLB instance, and three Master nodes in the backend provide the services.
- **Master node SSH IP address:** You can directly log on to the Master nodes by using SSH to perform routine maintenance for the cluster.

- **Service Access Domain:** Provides the services in the cluster with access domain name for testing. The service access domain name suffix is `<cluster_id>.<region_id>.alicontainer.com`.

1.3.6 Connect to a Kubernetes cluster by using kubectl

To connect to a Kubernetes cluster from a client computer, use the Kubernetes command line client *kubectl*.

Procedure

1. Download the latest kubectl client from the *Kubernetes release page*.
2. Install and set the kubectl client.

   For more information, see *Install and set kubectl*.
3. Configure the cluster credentials.

   You can use the `scp` command to safely copy the master node configurations from the `/etc/kubernetes/kube.conf` file on the master virtual machine of the Kubernetes cluster to the `$HOME/.kube/config` file (where the`kubectl` expected credentials reside) of the local computer.

   - If you select Password in the Login field when creating the cluster, copy the kubectl configuration file in the following method:

     ```
     mkdir $HOME/.kube
     scp root@<master-public-ip>:/etc/kubernetes/kube.conf $HOME/.kube/
     config
     ```
   - If you select Key Pair in the Login field when creating the cluster, copy the kubectl configuration file in the following method:

     ```
     mkdir $HOME/.kube
     scp -i [the storage path of the .pem private key file on the local
      machine] root@:/etc/kubernetes/kube.conf $HOME/.kube/config
     ```

   You can check the cluster `master-public-ip` on the cluster information page.

   a) Log on to the *Container Service console*.

   b) Under Kubernetes, click **Clusters** in the left-side navigation pane.

   c) Click **Manage** at the right of the cluster.

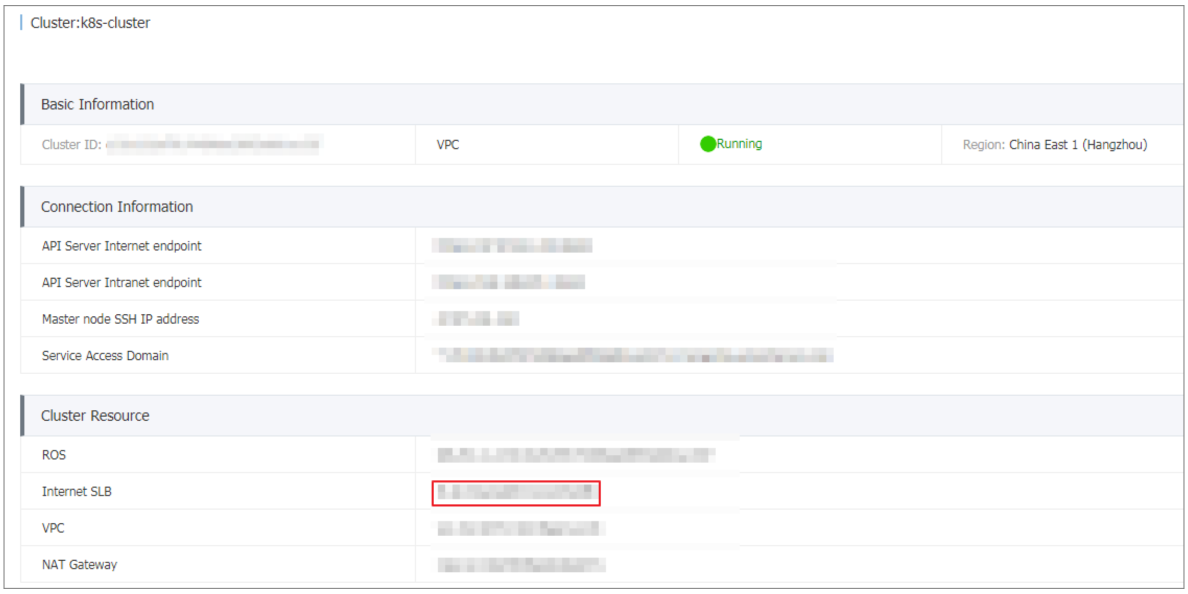
   In the **Connection Information** section, view the Master node SSH IP address.

1.3.7 Access Kubernetes clusters by using SSH

If you select not to enable SSH access for Internet when creating the Kubernetes cluster, you cannot access the Kubernetes cluster by using SSH or connect to the Kubernetes cluster by using kubectl. To access the cluster by using SSH after creating the cluster, manually bind Elastic IP (EIP) to the Elastic Compute Service (ECS) instance, configure security group rules, and open the SSH port (22).

Procedure

1. Log on to the *Container Service console*.
2. Under the Kubernetes menu, click **Clusters** in the left-side navigation pane.
3. Click **Manage** at the right of the cluster.
4. In **Cluster Resource**, click the ID of the Internet SLB. Then, you are redirected to the Instance Details page of your Internet Server Load Balancer instance.



Cluster:k8s-cluster

Basic Information

Cluster ID:	VPC	● Running	Region: China East 1 (Hangzhou)

Connection Information

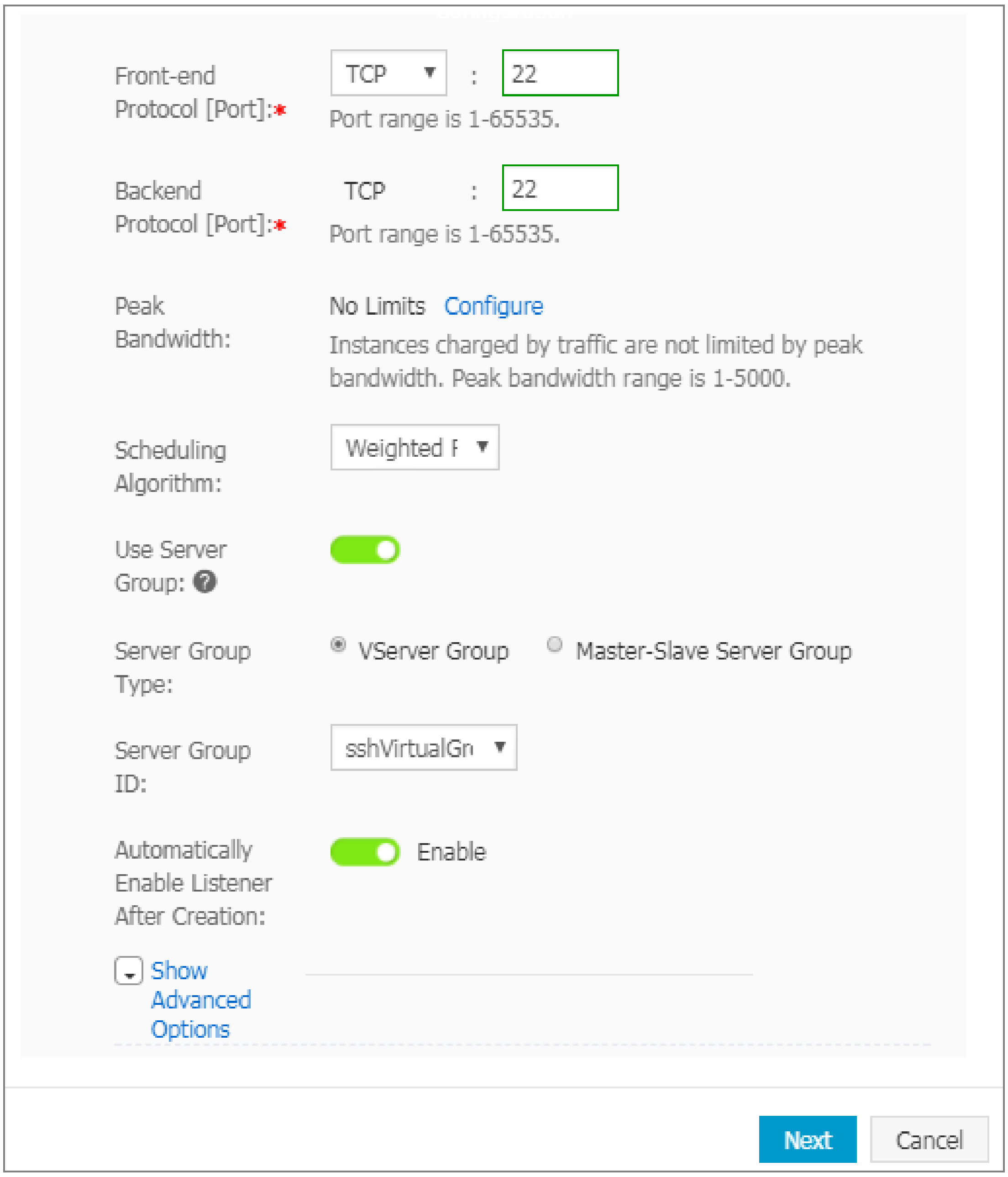
API Server Internet endpoint	
API Server Intranet endpoint	
Master node SSH IP address	
Service Access Domain	

Cluster Resource

ROS	
Internet SLB	
VPC	
NAT Gateway	

5. Select **Instances** > **Server Load Balancer**, and click **Add Listener**.
6. Add the SSH listening rule.

   a. **Front-end Protocol [Port]**: Select TCP and enter 22.
   b. **Backend Protocol [Port]**: Enter 22.
   c. Turn on the **Use Server Group** switch and select **VServer Group**.
   d. **Server Group ID**: Select **sshVirtualGroup**.
   e. Click **Next** and then click **Confirm** to create the listener.



Front-end Protocol [Port]:* TCP : 22
Port range is 1-65535.

Backend Protocol [Port]:* TCP : 22
Port range is 1-65535.

Peak Bandwidth: No Limits Configure
Instances charged by traffic are not limited by peak bandwidth. Peak bandwidth range is 1-5000.

Scheduling Algorithm: Weighted F

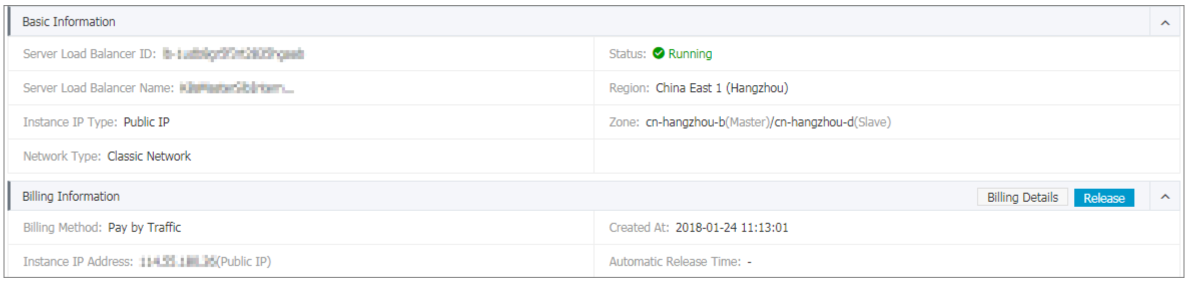
Use Server Group: ✓

Server Group Type: [✓ VServer Group] Master-Slave Server Group

Server Group ID: sshVirtualGr

Automatically Enable Listener After Creation: Enable

7. Then, you can use the Server Load Balancer instance IP address to access your cluster by using SSH.



Basic Information

Server Load Balancer ID: [redacted]	Status: ✓ Running
Server Load Balancer Name: [redacted]	Region: China East 1 (Hangzhou)
Instance IP Type: Public IP	Zone: cn-hangzhou-b(Master)/cn-hangzhou-d(Slave)
Network Type: Classic Network	

Billing Information

Billing Method: Pay by Traffic	Created At: 2018-01-24 11:13:01
Instance IP Address: [redacted](Public IP)	Automatic Release Time: -

1.3.8 Access Kubernetes clusters by using SSH key pairs

Alibaba Cloud Container Service allows you to log on to clusters by using SSH key pairs, which guarantees the security of SSH remote access.

Context

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane.
3. Click **Create Kubernetes Cluster** in the upper-right corner.
4. Select Key Pair in the Login field. Complete the other configurations. For more information, see *#unique_31*. Then, click **Create**.
   1. If you have created key pairs in the Elastic Compute Service (ECS) console, select a key pair from the Key Pair Name drop-down list.
   2. If you have no key pair, click **Create a new key pair** to create one in the ECS console. For more information, see *Create an SSH key pair*.
5. After the cluster is created, click **Manage** at the right of the cluster on the Cluster List page. View the **Master node SSH IP address** under Connection Information.
6. Download the *.pem* private key file. Complete the configurations based on your local operating system environment, such as Windows or Linux. For more information, see *Connect to a Linux instance by using an SSH key pair*. Take Linux as an example.
   a) Find the path where your downloaded .pem private key file is stored on your local machine. For example, `/root/xxx.pem`.
   b) Run the following command to modify the attributes of the private key file: `chmod 400 [path where the .pem private key file is stored on the local machine ]`. For example, `chmod 400 /root/xxx.pem`.
   c) Run the following command to connect to the cluster: `` `ssh -i [path where the .pem private key file is stored on the local machine] root@[master-public-ip] ``. Wherein, master-public-ip is the master node SSH IP address. For example, `ssh -i /root/xxx.pem root@10.10.10.100`.

1.3.9 Use a ServiceAccount token to access a managed Kubernetes cluster

This topic describes how to use a ServiceAccount token to access a managed Kubernetes cluster.

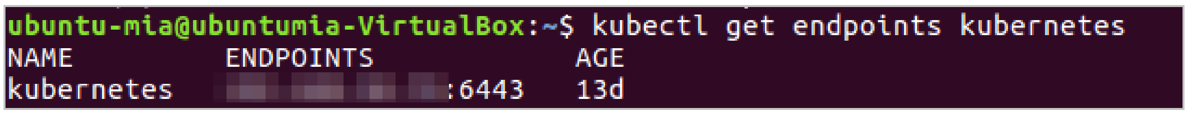
Context

- You have created a managed Kubernetes cluster. For more information, see *Create a managed Kubernetes cluster*.
- You have connected to the managed Kubernetes cluster by using kubectl, see *Connect to a Kubernetes cluster by using kubectl*.

Procedure

1. Run the following command to obtain the API server intranet endpoint:

```
$ kubectl get endpoints kubernetes
```



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

2. Create a file named *`kubernetes-public-service.yaml`* and set the **`ip`** parameter to the intranet endpoint obtained in step 1.

```
kind: Service
apiVersion: v1
metadata:
  name: kubernetes-public
spec:
  type: LoadBalancer
  ports:
  - name: https
    port: 443
    protocol: TCP
    targetPort: 6443
---
apiVersion: v1
kind: Endpoints
metadata:
  name: kubernetes-public
  namespace: default
subsets:
- addresses:
  - ip: <API Service address>  #Set this parameter to the intranet
endpoint obtained in step 1.
  ports:
  - name: https
    port: 6443
    protocol: TCP
```

3. Run the following command to deploy the API server Internet endpoint:

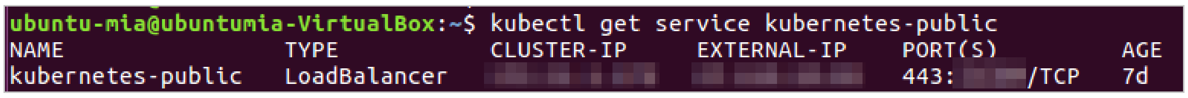
```
$ kubectl apply -f kubernetes-public-service.yaml
```

4. Run the following command to obtain the Internet SLB address, namely, **`EXTERNAL-IP`**:

```
$ kubectl get service name
```

Note:

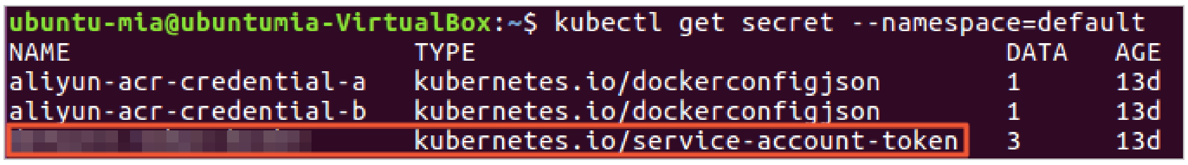
The *name* parameter in the command and the *name* parameter in the *`kubernetes-public-service.yaml`* file of step 2 must be set to the same value. In this example, this parameter is set to `kubernetes-public`.



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

5. Run the following command to view the corresponding secret of the ServiceAccount (in this example, the *namespace* parameter is set to default):

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



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

6. Run the following command to obtain a token value:

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

Note:

The *namespace* parameter in this command and the *namespace* parameter in step 5 must be set to the same value.

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

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

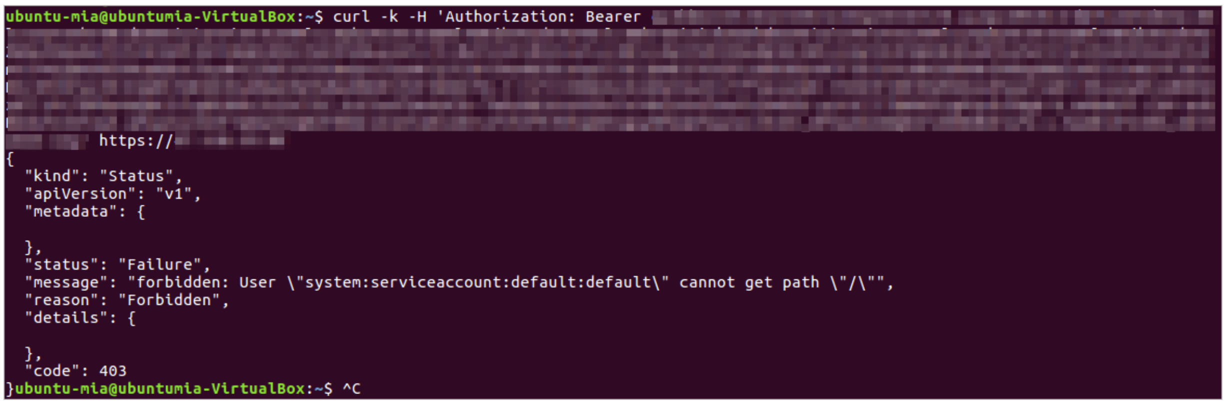
Note:

- The value of **`token`** is the token value obtained in step 6.

- The value of **`service-ip`** is the Internet SLB address obtained in step 4, that is, **`EXTERNAL-IP`**.

Result

After you run the command, the following message is displayed, indicating that you have connected to the cluster.

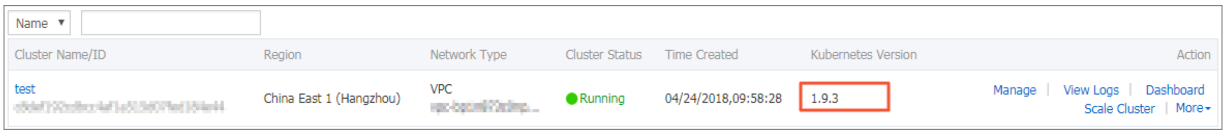


```
ubuntu-mia@ubuntumia-VirtualBox:~$ curl -k -H 'Authorization: Bearer [redacted]
[redacted]
[redacted] https://[redacted]
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "forbidden: User \"system:serviceaccount:default:default\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {

  },
  "code": 403
}ubuntu-mia@ubuntumia-VirtualBox:~$ ^C
```

1.3.10 Upgrade a Kubernetes cluster

You can upgrade the version of your Kubernetes cluster in the Container Service console.

On the cluster list page, you can view the version of your Kubernetes cluster.



Cluster Name/ID	Region	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
test [redacted]	China East 1 (Hangzhou)	VPC [redacted]	● Running	04/24/2018,09:58:28	1.9.3	Manage | View Logs | Dashboard | Scale Cluster | More▾

Prerequisites

- Make sure that your host can access the Internet so that the system can download the required software package.
- A cluster upgrade may fail. We recommend that you create a snapshot for your cluster to guarantee your data security before upgrading the cluster. For more information, see *Create a snapshot*.
- If you are upgrading a Kubernetes cluster of version number V1.8.1 or V1.8.4 to V1.9.3, all cluster pods will be restarted. This means that applications running on the cluster will be affected. If you are upgrading a Kubernetes cluster version of a different number, cluster applications will not be affected. However, if a cluster application is highly dependent on the API server, the application may be temporarily affected by the upgrade.

- OSS volumes will be re-mounted to the cluster because the network is reset during the cluster upgrade. Therefore, you need to re-create the pods that use the OSS volumes after the upgrade.

Preparations

You must make sure that your cluster is in healthy status before the upgrade.

You must to log on to a Master node. For more information, see *Access Kubernetes clusters by using SSH* and *Connect to a Kubernetes cluster by using kubectl*.

1. Run the `kubectl get cs` command to verify that all cluster modules are healthy.

```
NAME                     STATUS    MESSAGE              ERROR
 scheduler                Healthy   ok
 controller-manager       Healthy   ok
 etcd-0                   Healthy   {"health": "true"}
 etcd-1                   Healthy   {"health": "true"}
 etcd-2                   Healthy   {"health": "true"}
```

2. Run the `kubectl get nodes` command to verify that all nodes are ready.

Note:

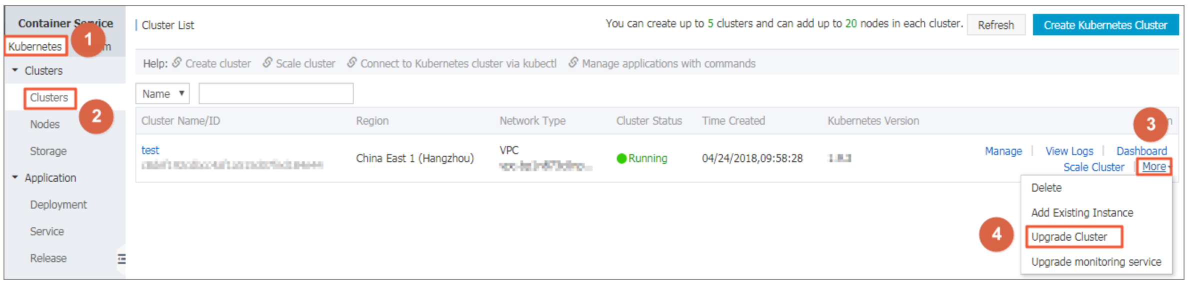
All nodes must be in ready status only.

```
kubectl get nodes
 NAME                   STATUS    ROLES     AGE       VERSION
 cn-shanghai.i-xxxxxx   Ready     master    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     <none>    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     <none>    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     <none>    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     master    38d       v1.9.3
 cn-shanghai.i-xxxxxx   Ready     master    38d       v1.9.3
```

If a node is abnormal, you can fix it on your own or you can submit a ticket to ask for technical support from Alibaba Cloud.

Procedure

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


Container Service
Kubernetes
1
Clusters
Clusters
2
Nodes
Storage
Application
Deployment
Service
Release
Cluster List
You can create up to 5 clusters and can add up to 20 nodes in each cluster.
Refresh
Create Kubernetes Cluster
Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands
Name
3
Manage | View Logs | Dashboard
Scale Cluster | More
Delete
Add Existing Instance
4
Upgrade Cluster
Upgrade monitoring service

Cluster Name/ID	Region	Network Type	Cluster Status	Time Created	Kubernetes Version
test [redacted]	China East 1 (Hangzhou)	VPC [redacted]	Running	04/24/2018,09:58:28	[redacted]

4. In the displayed dialog box, click **Upgrade**.

   The system starts to upgrade the Kubernetes version.

After the upgrade is completed, you can view the Kubernetes version of the cluster on the cluster list page and verify that the upgrade is successful.

1.3.11 Notice about fixed Kubernetes vulnerability 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.

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 permissions 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 *Upgrade a Kubernetes cluster*.

- 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**.

This vulnerability does not affect Serverless Kubernetes clusters. Serverless Kubernetes was upgraded before the vulnerability occurred.

1.3.12 Scale out or in a cluster

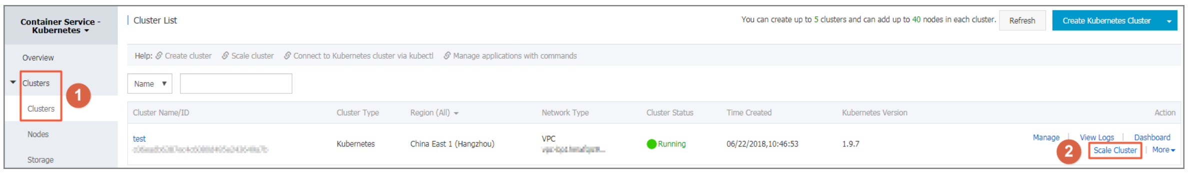
In the Container Service console, you can scale out or scale in the worker nodes of a Kubernetes cluster according to your actual business requirements.

Context

- Currently, Container Service does not support scaling in or out master nodes in a cluster.
- Container Service only supports scaling in worker nodes that are added when you create or scale out the cluster. These worker nodes cannot be removed either by using the **`kubectl delete`** command or through the console. Worker nodes that are added to the cluster through *Add an existing ECS instance* cannot be scaled in.
- When you scale in a cluster, the worker nodes are removed from the cluster in the order that they are added after you scale out the cluster.
- You must have more than 1 node that is not manually added to perform scaling in.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane.
3. Click **Scale Cluster** at the right of the cluster.

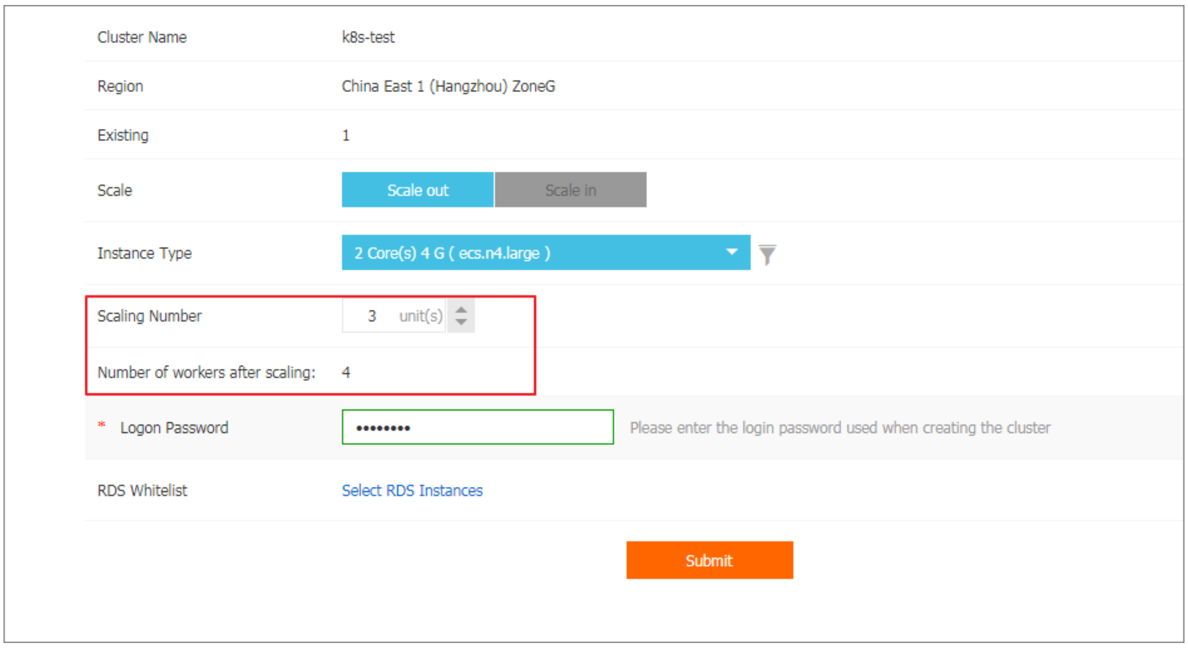

Container Service - Kubernetes
Overview
Clusters
Clusters
1
Nodes
Storage
Cluster List
Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands
Name
Refresh
Create Kubernetes Cluster
2

You can create up to 5 clusters and can add up to 40 nodes in each cluster.

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
test [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC [redacted]	Running	06/22/2018,10:46:53	1.9.7	Manage | View Logs | Dashboard Scale Cluster | More

4. Select **Scale out** or **Scale in** in the Scale field and then configure the number of worker nodes.

   In this example, scale out the cluster to change the number of worker nodes from one to four.



Cluster Name	k8s-test
Region	China East 1 (Hangzhou) ZoneG
Existing	1
Scale	Scale out
Instance Type	2 Core(s) 4 G (ecs.n4.large)
Scaling Number	3 unit(s)
Number of workers after scaling:	4
* Logon Password	•••••••• Please enter the login password used when creating the cluster
RDS Whitelist	Select RDS Instances

5. Enter the logon password of the node.

Note:

Make sure this password is the same as the one you entered when creating the cluster because you have to log on to the Elastic Compute Service (ECS) instance to copy the configuration information in the upgrade process.

6. Click **Submit**.

What's next

After completing the cluster, click **Clusters** > **Node** in the left-side navigation pane. On the Node List page, you can see that the current number of worker nodes is changed to 4.

1.3.13 Cluster auto scaling

Configure a cluster to auto scale according to the cluster load.

Prerequisites

You have created a Kubernetes cluster successfully. For how to create a Kubernetes cluster, see *Create a Kubernetes cluster*.

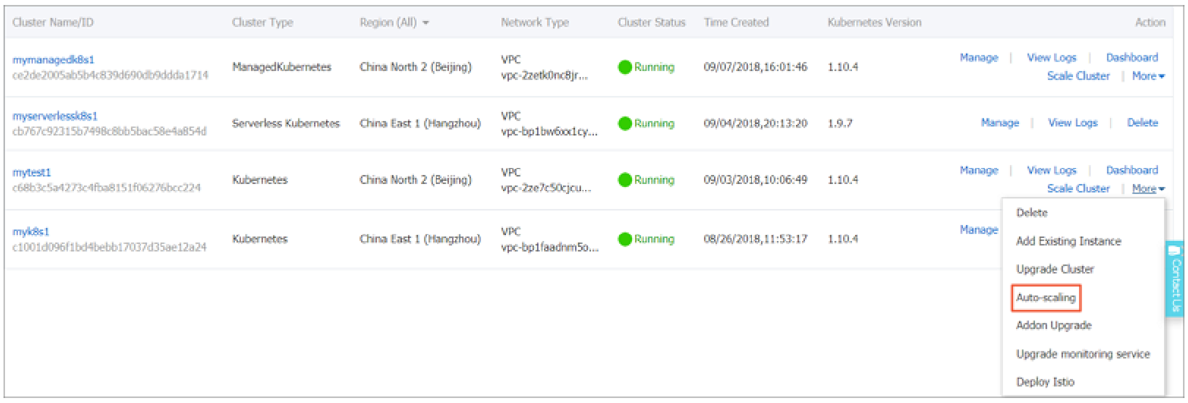
Background

Cluster auto scaling is different from *Scale out or in a cluster* that is based on resource thresholds. After auto scaling is configured for a cluster, the cluster automatically scales out or scales in when the cluster load reaches the configured value.

Procedure

Enable cluster auto scaling

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane.
3. Select a cluster and click **More** > **Auto Scaling** in the Action column.

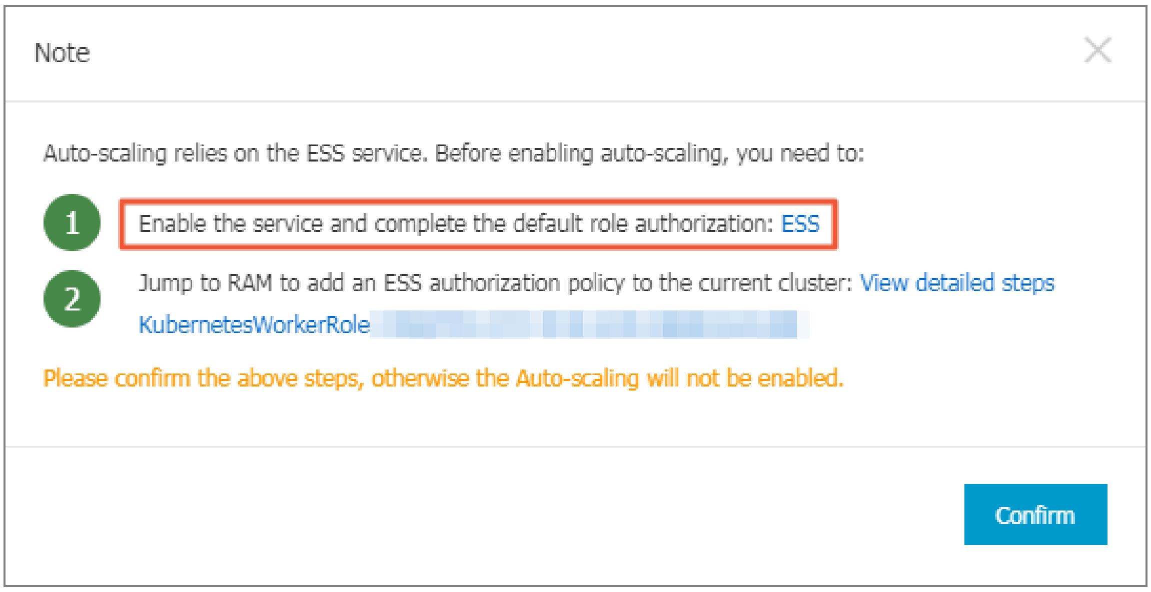


Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
mymanagedk8s1 ce2de2005ab5b4c839d690db9ddda1714	ManagedKubernetes	China North 2 (Beijing)	VPC vpc-2zetk0nc8jr...	● Running	09/07/2018,16:01:46	1.10.4	Manage | View Logs | Dashboard Scale Cluster | More
myserverlessk8s1 cb767c92315b7498c8bb5bac58e4a854d	Serverless Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1bw6xx1cy...	● Running	09/04/2018,20:13:20	1.9.7	Manage | View Logs | Delete
mytest1 c68b3c5a4273c4fba8151f06276bcc224	Kubernetes	China North 2 (Beijing)	VPC vpc-2ze7c50cjcu...	● Running	09/03/2018,10:06:49	1.10.4	Manage | View Logs | Dashboard Scale Cluster | More
myk8s1 c1001d096f1bd4bebb17037d35ae12a24	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1faadnm5o...	● Running	08/26/2018,11:53:17	1.10.4	Manage

Authorization

- **Activate Auto Scaling service**

1. Click the first link in the displayed dialog box.



Note

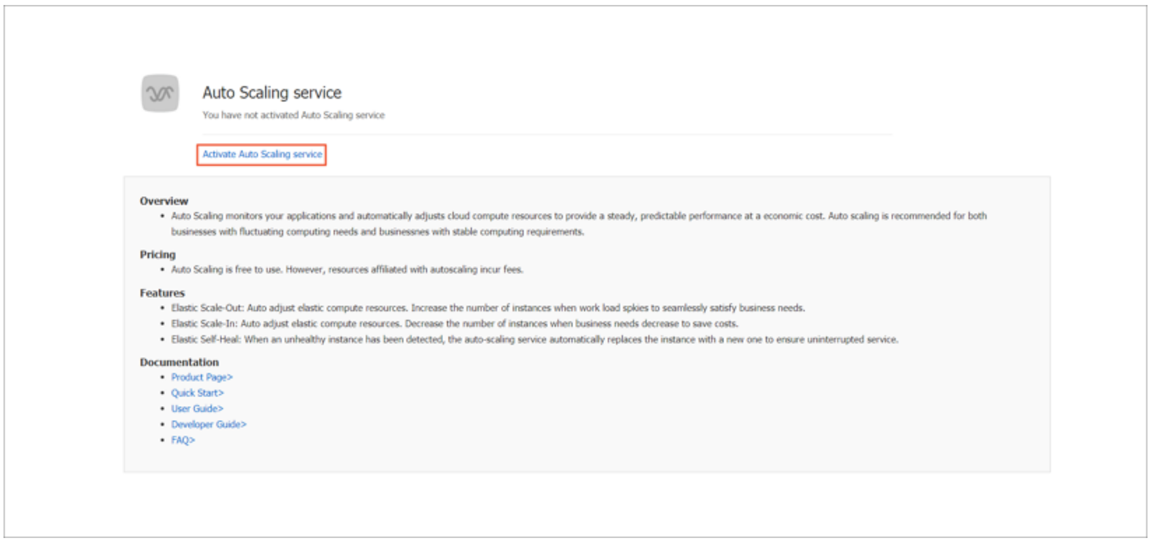
Auto-scaling relies on the ESS service. Before enabling auto-scaling, you need to:

1 Enable the service and complete the default role authorization: ESS

2 Jump to RAM to add an ESS authorization policy to the current cluster: View detailed steps
KubernetesWorkerRole [redacted]

Please confirm the above steps, otherwise the Auto-scaling will not be enabled.

2. Click **Activate Auto Scaling service**.



Auto Scaling service

You have not activated Auto Scaling service

Activate Auto Scaling service

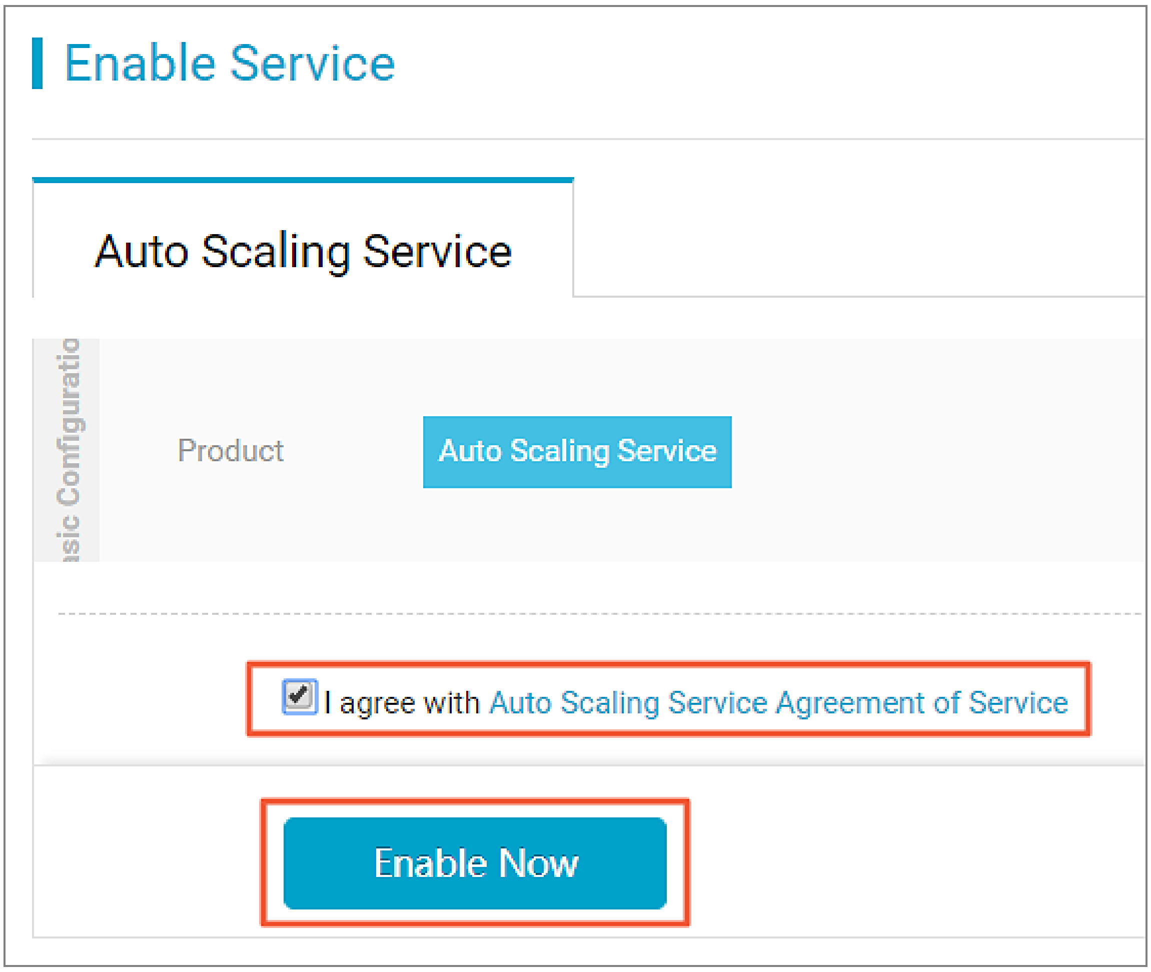
Overview
- Auto Scaling monitors your applications and automatically adjusts cloud compute resources to provide a steady, predictable performance at a economic cost. Auto scaling is recommended for both businesses with fluctuating computing needs and businessnes with stable computing requirements.

Pricing
- Auto Scaling is free to use. However, resources affiliated with autoscaling incur fees.

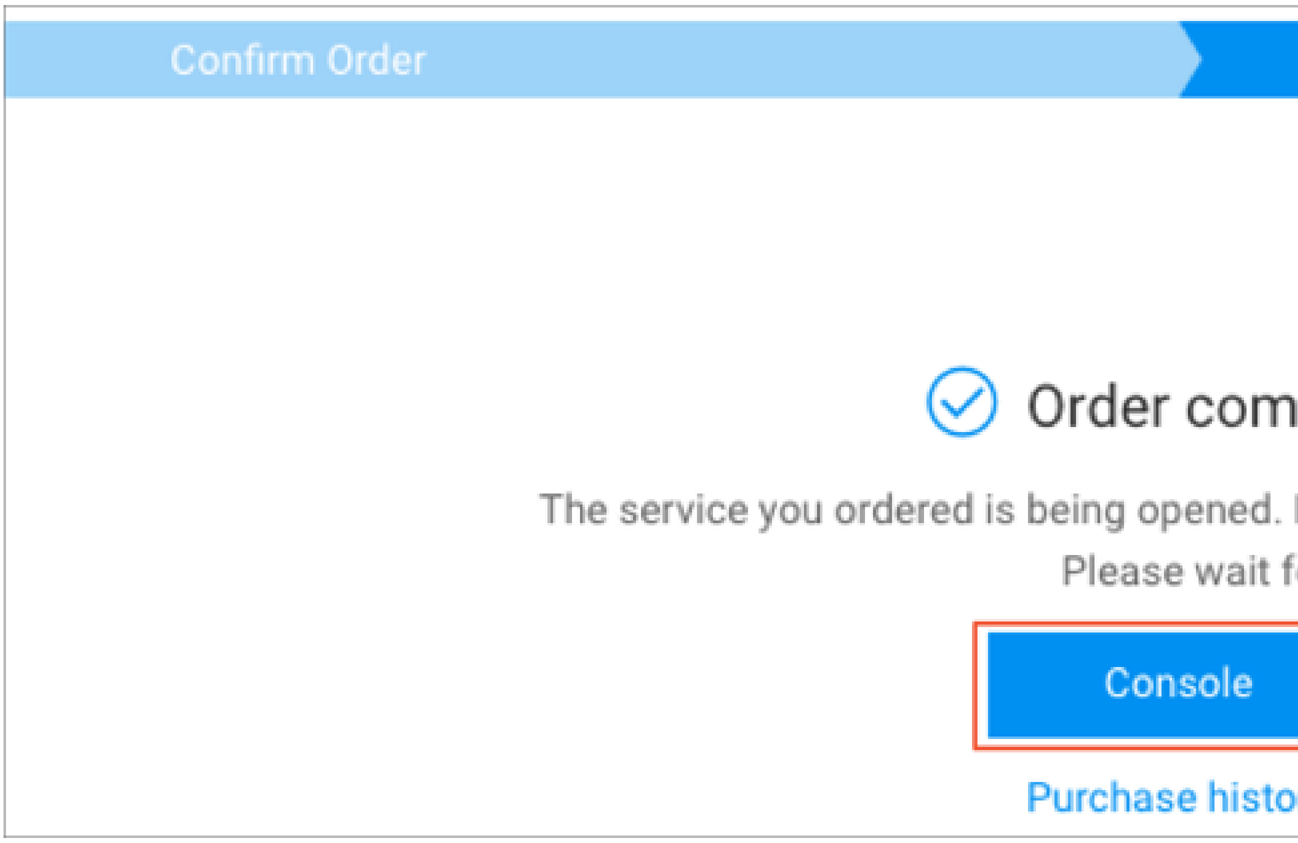
Features
- Elastic Scale-Out: Auto adjust elastic compute resources. Increase the number of instances when work load spkies to seamlessly satisfy business needs.
- Elastic Scale-In: Auto adjust elastic compute resources. Decrease the number of instances when business needs decrease to save costs.
- Elastic Self-Heal: When an unhealthy instance has been detected, the auto-scaling service automatically replaces the instance with a new one to ensure uninterrupted service.

Documentation
- Product Page>
- Quick Start>
- User Guide>
- Developer Guide>
- FAQ>

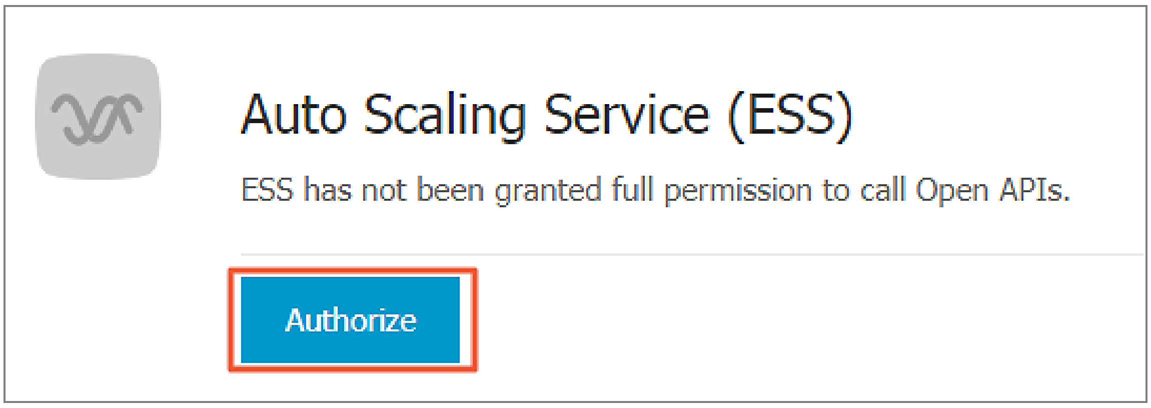
3. Select the **I agree with Auto Scaling Service Agreement of Service** check box and click **Enable Now**.



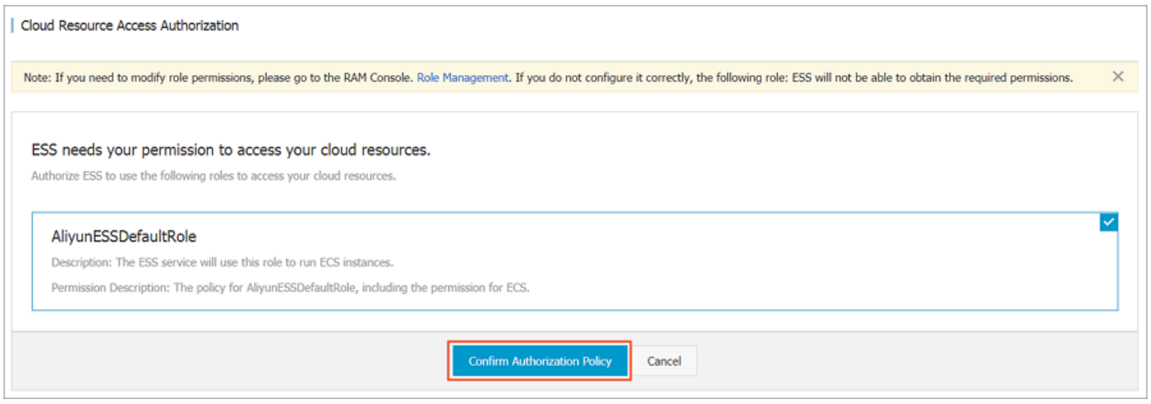
4. After the service is activated, click **Console**.



5. Click **Authorize**. Configure permissions for accessing cloud resources on the **Cloud Resource Access Authorization** page.


Auto Scaling Service (ESS)
ESS has not been granted full permission to call Open APIs.
Authorize

6. Click **Confirm Authorization Policy**.


Cloud Resource Access Authorization

Note: If you need to modify role permissions, please go to the RAM Console. Role Management. If you do not configure it correctly, the following role: ESS will not be able to obtain the required permissions.

ESS needs your permission to access your cloud resources.

Authorize ESS to use the following roles to access your cloud resources.

AliyunESSDefaultRole

Description: The ESS service will use this role to run ECS instances.

Permission Description: The policy for AliyunESSDefaultRole, including the permission for ECS.

Confirm Authorization Policy Cancel

Expected result

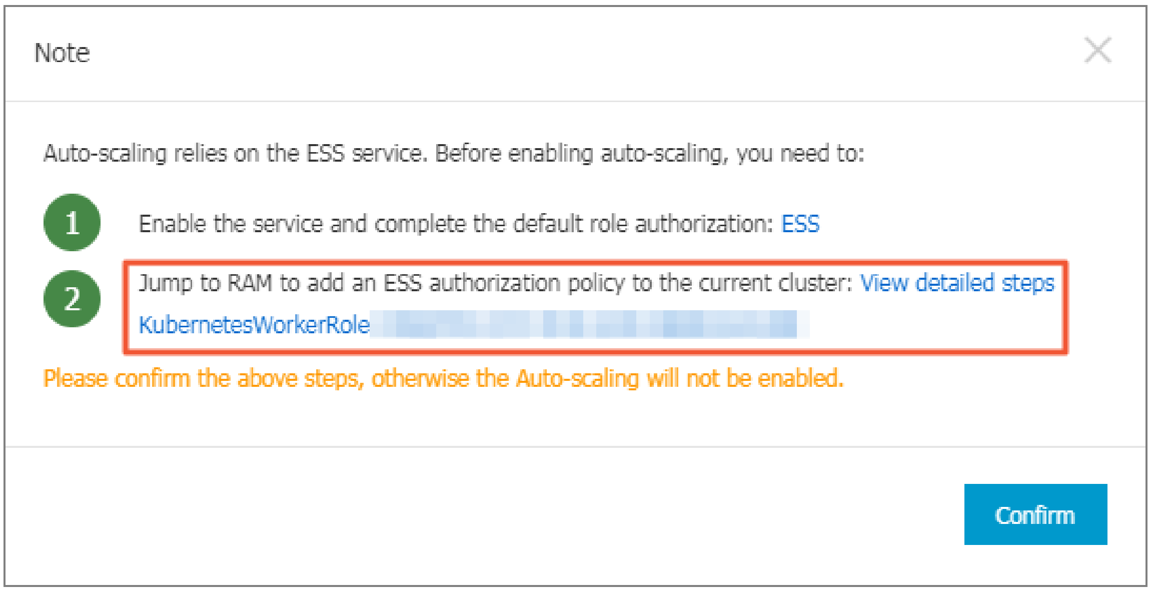
When the page automatically jumps to the elastic scaling console, the authorization succeeds.

Closes the page and continues to configure **Authorize roles**.

- **Authorize a role.**

1. Click the second link in the displayed dialog box.

Note:
Use the primary account to log on to the console.


Note

Auto-scaling relies on the ESS service. Before enabling auto-scaling, you need to:

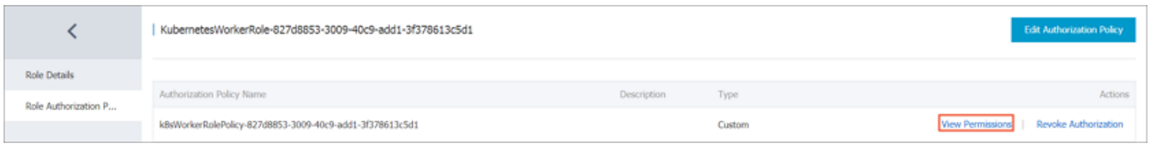
1 Enable the service and complete the default role authorization: ESS

2 Jump to RAM to add an ESS authorization policy to the current cluster: View detailed steps KubernetesWorkerRole [redacted]

Please confirm the above steps, otherwise the Auto-scaling will not be enabled.

Confirm

2. Select the target authorization policy and click **View Permissions** in the Action column.

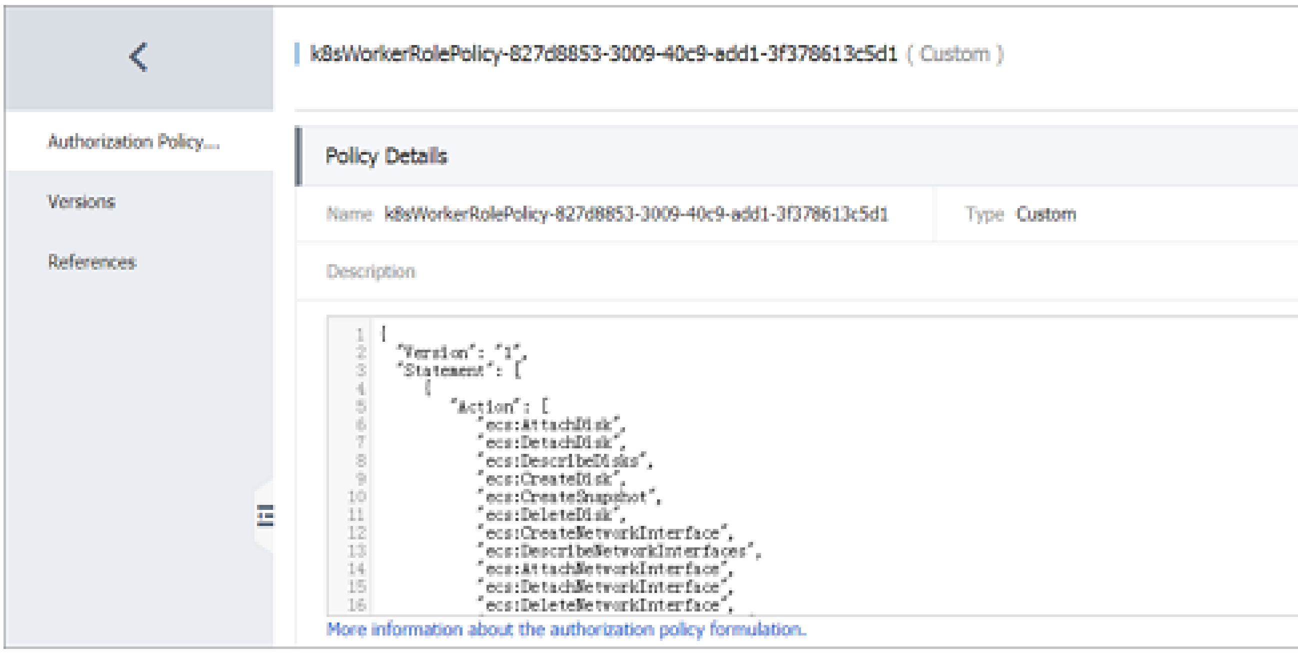

KubernetesWorkerRole-827d8853-3009-40c9-add1-3f378613c5d1 Edit Authorization Policy

Role Details

Role Authorization P...

Authorization Policy Name	Description	Type	Actions
k8sWorkerRolePolicy-827d8853-3009-40c9-add1-3f378613c5d1		Custom	View Permissions | Revoke Authorization

3. Click **Modify Authorization Policy** in the upper-right corner of the page.


k8sWorkerRolePolicy-827d8853-3009-40c9-add1-3f378613c5d1 (Custom)

Authorization Policy...

Versions

References

Policy Details

Name k8sWorkerRolePolicy-827d8853-3009-40c9-add1-3f378613c5d1 Type Custom

Description

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "ecs:AttachDisk",
        "ecs:DetachDisk",
        "ecs:DescribeDisks",
        "ecs:CreateDisk",
        "ecs:CreateSnapshot",
        "ecs:DeleteDisk",
        "ecs:CreateNetworkInterface",
        "ecs:DescribeNetworkInterfaces",
        "ecs:AttachNetworkInterface",
        "ecs:DetachNetworkInterface",
        "ecs:DeleteNetworkInterface",
```

More information about the authorization policy formulation.

4. In the `Action` field of **Policy content**, enter the following:

```
"ess:Describe*",
"ess:CreateScalingRule",
"ess:ModifyScalingGroup",
"ess:RemoveInstances",
"ess:ExecuteScalingRule",
"ess:ModifyScalingRule",
"ess:DeleteScalingRule",
"ecs:DescribeInstanceTypes",
"ess:DetachInstances"
```

Note:

Add a comma (,) to the last line in the `Action` field before entering the preceding content.

5. Click **Modify Authorization**.

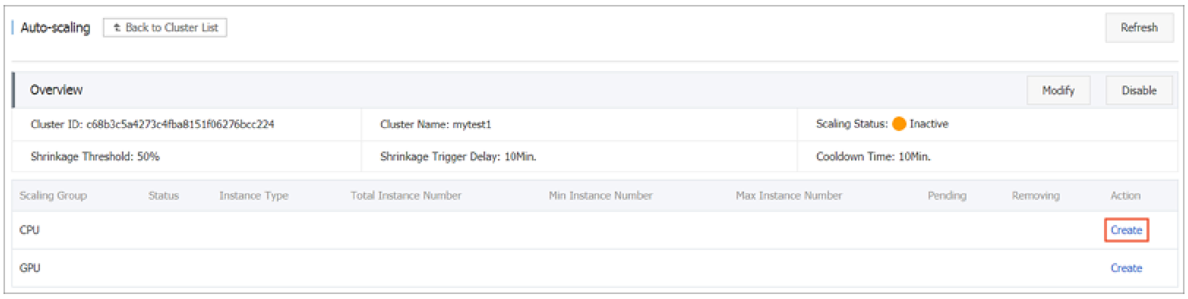
Configure cluster auto scaling

1. On the **Auto-scaling** page, configure the following parameters:

Configuration	Description
Cluster	The target cluster name.
Shrinkage Threshold	The ratio of the amount of resources requested by the cluster load to the amount of cluster resources. When the amount of resource requested by the cluster load is less than or equal to the configured shrinkage

Configuration	Description
	threshold, the cluster automatically shrinks. The default is 50%.
Shrinkage Trigger Delay	When the configured shrinkage threshold is reached and the configured shrinkage trigger delay expires, the cluster starts to shrinks. Unit: minute The default is 10 minutes.
Cooldown Time	After cluster expansion or shrinkage, the cooldown time starts to count. During the cooldown time, adding nodes to or removing nodes from the cluster does not trigger the cluster to expand or shrink again. The default is 10 minutes.

2. Select a resource type (CPU or GPU) to be scaled, click **Create** in the Action column.



Cluster ID: c68b3c5a4273c4fba8151f06276bcc224
Cluster Name: mytest1
Scaling Status: Inactive
Shrinkage Threshold: 50%
Shrinkage Trigger Delay: 10Min.
Cooldown Time: 10Min.

Scaling Group	Status	Instance Type	Total Instance Number	Min Instance Number	Max Instance Number	Pending	Removing	Action
CPU								Create
GPU								Create

On the **Scaling Group Config** page, configure the following parameters to create a scaling group:

Configuration	Description
Region	The region to which the created scaling group is deployed. This region must be consistent with the region of the cluster in which the scaling group resides. This region cannot be changed.
Zone	The zone of the created scaling group.
VPC	The network of the created scaling group, which must be consistent with the network of the cluster in which the scaling group resides.

Configure worker nodes

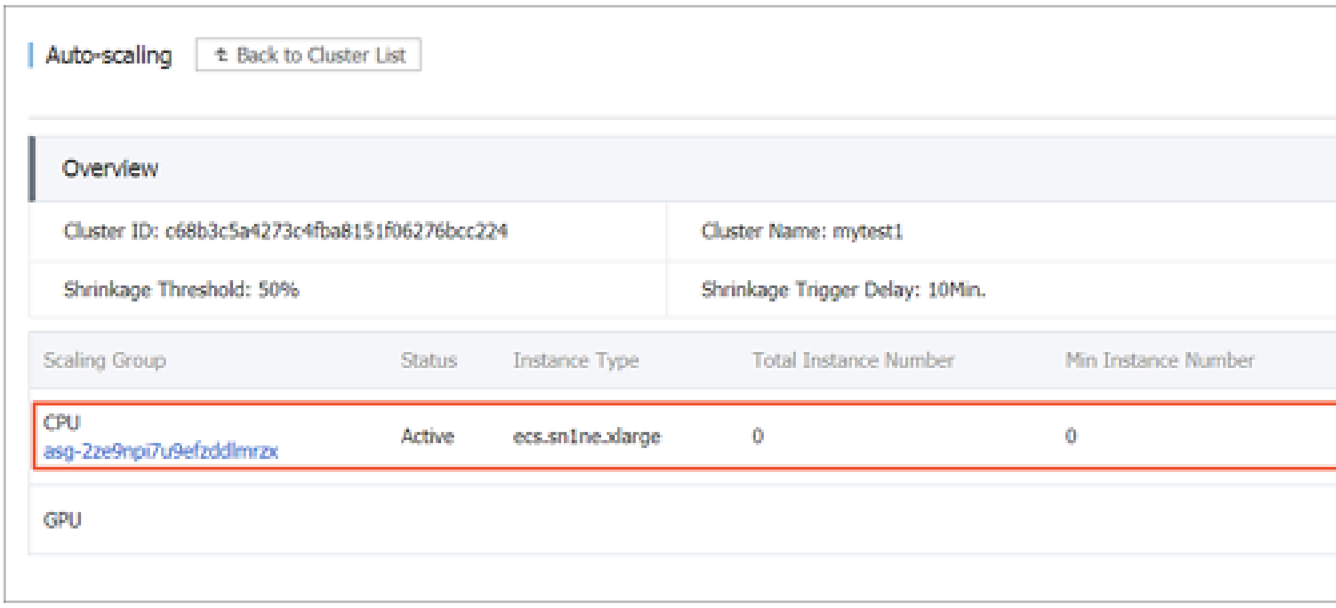
Configuration	Description
Instance Type	Types of instances in the scaling group.

Configuration	Description
System Disk	The system disk of the scaling group.
Attach Data Disk	Whether or not to mount a data disk when you create a scaling group. By default, no data disk is not mounted.
Instance Quantity	The number of instances contained by the scaling group. **Note:** • Existing instances are not included. • By default, the minimum number of instances is 0. When the number of instances exceeds 0, the cluster adds an instance to the scaling group and adds the instance into the Kubernetes cluster in which the scaling group resides by default.
Key Pair	The key pair used to log on to the scaled node. You can create a new key pair on the Elastic Compute Service (ECS) console. **Note:** Only key pair logon is supported currently.
RDS whitelist	The Relational Database Service (RDS) instance that can be accessed by a scaled node.

3. Click **OK** to create a scaling group.

Expected result

- A scaling group is displayed under CPU on the **Auto-scaling** page.



Cluster ID: c68b3c5a4273c4fba8151f06276bcc224

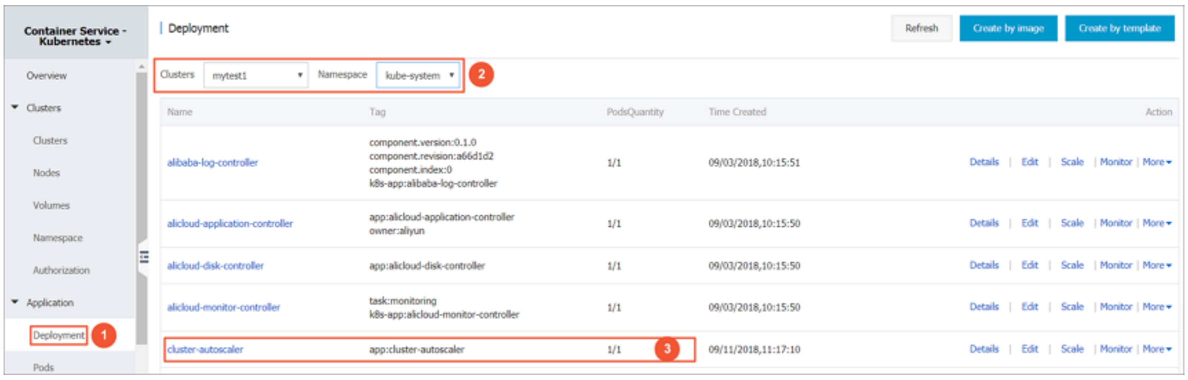
Cluster Name: mytest1

Shrinkage Threshold: 50%

Shrinkage Trigger Delay: 10Min.

Scaling Group	Status	Instance Type	Total Instance Number	Min Instance Number
CPU asg-2ze9npi7u9efzddlmrzx	Active	ecs.sn1ne.xlarge	0	0
GPU				

- 1. Click **Application** > **Deployment** in the left-side navigation pane.
  2. Select the target cluster and the kube-system namespace, you can view the created component named cluster-autoscaler.



Clusters mytest1 Namespace kube-system

Name	Tag	PodsQuantity	Time Created	Action
alibaba-log-controller	component.version:0.1.0 component.revision:a66d1d2 component.index:0 k8s-app:alibaba-log-controller	1/1	09/03/2018,10:15:51	Details | Edit | Scale | Monitor | More
alicloud-application-controller	app:alicloud-application-controller owner:aliyun	1/1	09/03/2018,10:15:50	Details | Edit | Scale | Monitor | More
alicloud-disk-controller	app:alicloud-disk-controller	1/1	09/03/2018,10:15:50	Details | Edit | Scale | Monitor | More
alicloud-monitor-controller	task:monitoring k8s-app:alicloud-monitor-controller	1/1	09/03/2018,10:15:50	Details | Edit | Scale | Monitor | More
cluster-autoscaler	app:cluster-autoscaler	1/1	09/11/2018,11:17:10	Details | Edit | Scale | Monitor | More

1.3.14 Delete a cluster

In the Container Service console, you can delete clusters that are no longer in use.

Procedure

1. Log on to the *Container Service console*.
2. Under Container Service Kubernetes, click **Clusters** in the left-side navigation pane.
3. Click **More** > **Delete** and select the target cluster.

What's next

Failed to delete a cluster

If you manually add some resources under the resources created by Resource Orchestration Service (ROS), ROS does not have permissions to delete the manually added resources. For example, manually add a VSwitch under the Virtual Private Cloud (VPC) created by ROS. ROS fails to process this VPC when deleting the Kubernetes resources and then the cluster fails to be deleted.

Container Service allows you to force delete the cluster. You can force delete the cluster record and ROS stack if the cluster fails to be deleted. However, you must release the created resources manually.

The cluster status is Failed if the cluster fails to be deleted.

Click**More** > **Delete** in the displayed dialog box, you can see the resource that failed to delete, check force delete, and click **OK**. The cluster and ROS resource can be deleted.

Note:

You must manually release the resources that failed to be deleted. For information on how to troubleshoot the problem with resources that cannot be released, see *#unique_44*.

1.4 Nodes

1.4.1 Add an existing ECS instance

You can add existing Elastic Compute Service (ECS) instances to a created Kubernetes cluster. Currently, 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 first.

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.

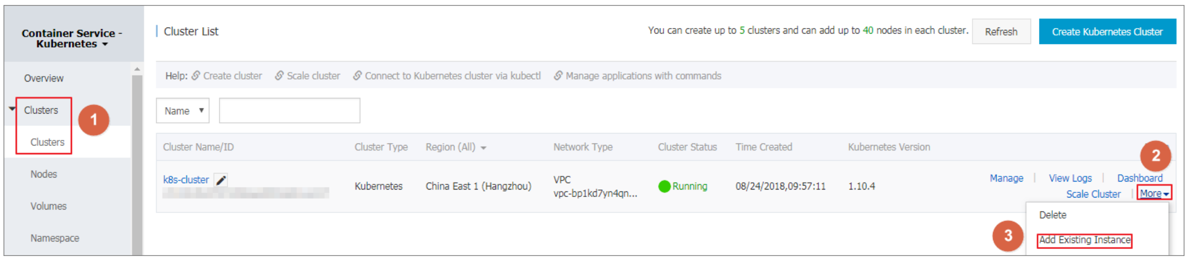
- When adding an existing instance, make sure that your instance has an Elastic IP (EIP) for the VPC network type, or the corresponding VPC is already configured with the NAT gateway. In short, make sure the corresponding node can access public network normally. Otherwise, the ECS instance fails to be 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. Under Kubernetes, click **Clusters** in the left-side navigation pane.
3. Select the target cluster and click **More** > **Add Existing Instance**.

   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.

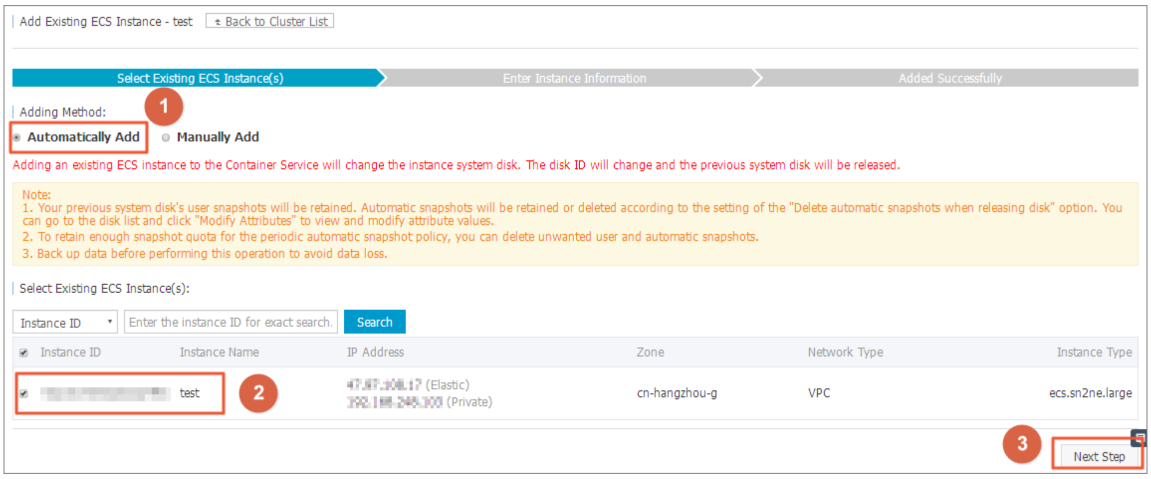


You can create up to 5 clusters and can add up to 40 nodes in each cluster.

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Time Created	Kubernetes Version
k8s-cluster [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1kd7yn4qn...	Running	08/24/2018,09:57:11	1.10.4

4. Select Automatically Add 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**.


Add Existing ECS Instance - test | Back to Cluster List

Select Existing ECS Instance(s) > Enter Instance Information > Added Successfully

Adding Method:
[✓ Automatically Add] Manually Add

Adding an existing ECS instance to the Container Service will change the instance system disk. The disk ID will change and the previous system disk will be released.

Note:
1. Your previous system disk's user snapshots will be retained. Automatic snapshots will be retained or deleted according to the setting of the "Delete automatic snapshots when releasing disk" option. You can go to the disk list and click "Modify Attributes" to view and modify attribute values.
2. To retain enough snapshot quota for the periodic automatic snapshot policy, you can delete unwanted user and automatic snapshots.
3. Back up data before performing this operation to avoid data loss.

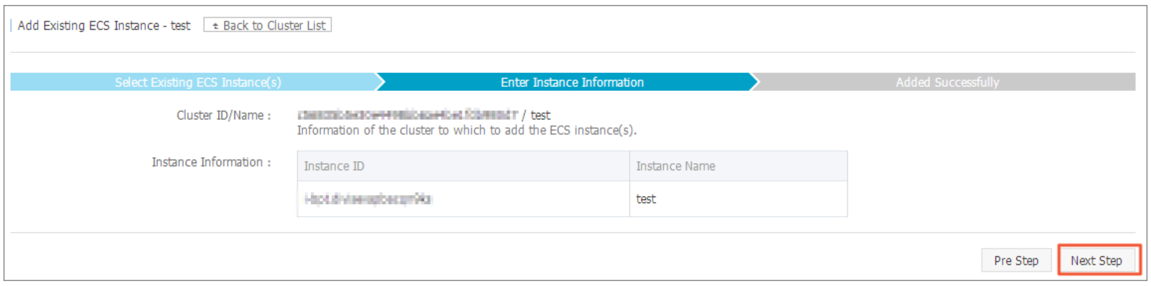
Select Existing ECS Instance(s):

Instance ID | Enter the instance ID for exact search. | Search

✓	Instance ID	Instance Name	IP Address	Zone	Network Type	Instance Type
✓	[redacted]	test	[redacted] (Elastic) [redacted] (Private)	cn-hangzhou-g	VPC	ecs.sn2ne.large

Next Step

b) Enter the instance information, set the logon password, and then click **Next Step**.


Add Existing ECS Instance - test | Back to Cluster List

Select Existing ECS Instance(s) > Enter Instance Information > Added Successfully

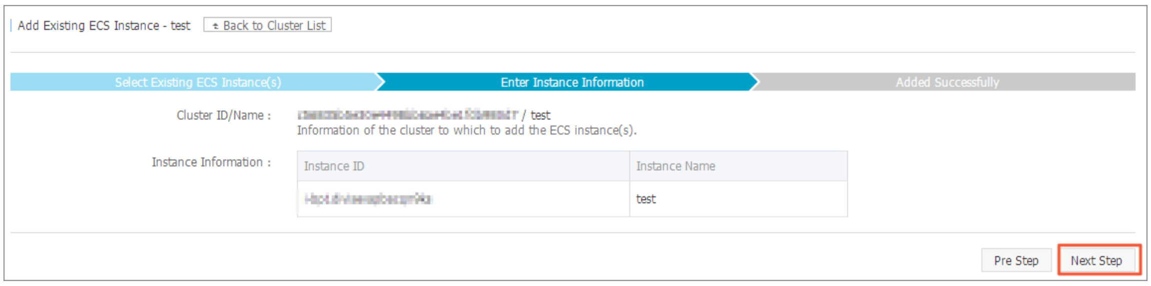
Cluster ID/Name : [redacted] / test
Information of the cluster to which to add the ECS instance(s).

Instance Information :

Instance ID	Instance Name
[redacted]	test

Pre Step | Next Step

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


Add Existing ECS Instance - test | Back to Cluster List

Select Existing ECS Instance(s) > Enter Instance Information > Added Successfully

Cluster ID/Name : [redacted] / test
Information of the cluster to which to add the ECS instance(s).

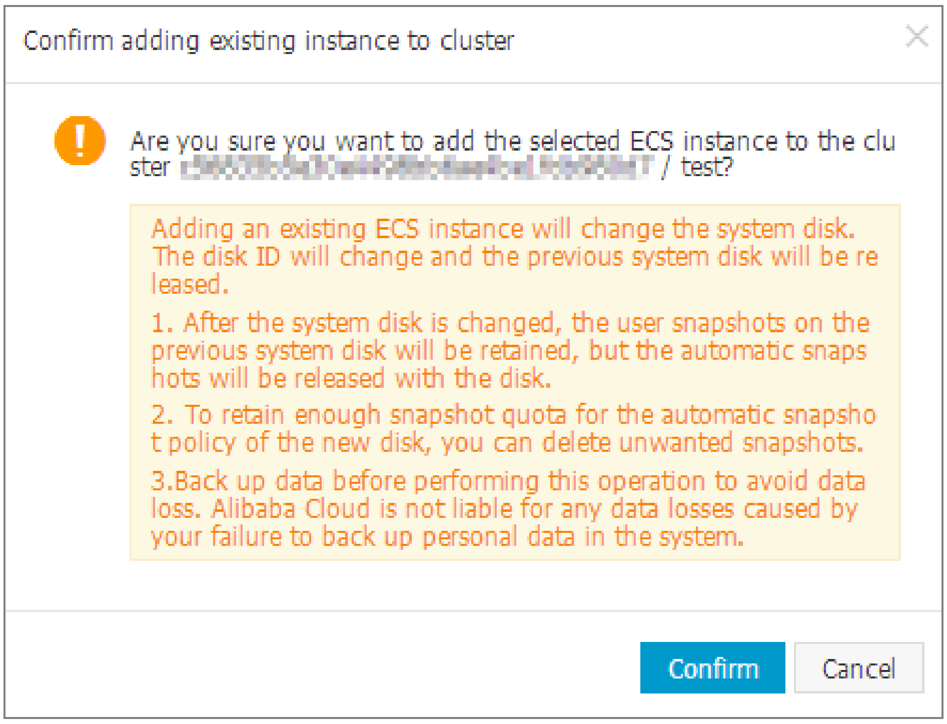
Instance Information :

Instance ID	Instance Name
[redacted]	test

Pre Step | Next Step

5. Optional: You can also select Manually Add 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.



Confirm adding existing instance to cluster

Are you sure you want to add the selected ECS instance to the cluster [redacted] / test?

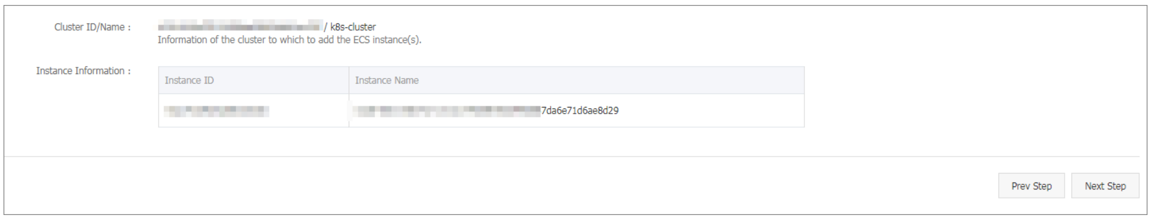
Adding an existing ECS instance will change the system disk. The disk ID will change and the previous system disk will be released.

1. After the system disk is changed, the user snapshots on the previous system disk will be retained, but the automatic snapshots will be released with the disk.

2. To retain enough snapshot quota for the automatic snapshot policy of the new disk, you can delete unwanted snapshots.

3.Back up data before performing this operation to avoid data loss. Alibaba Cloud is not liable for any data losses caused by your failure to back up personal data in the system.

b) Confirm the information and then click **Next Step**.



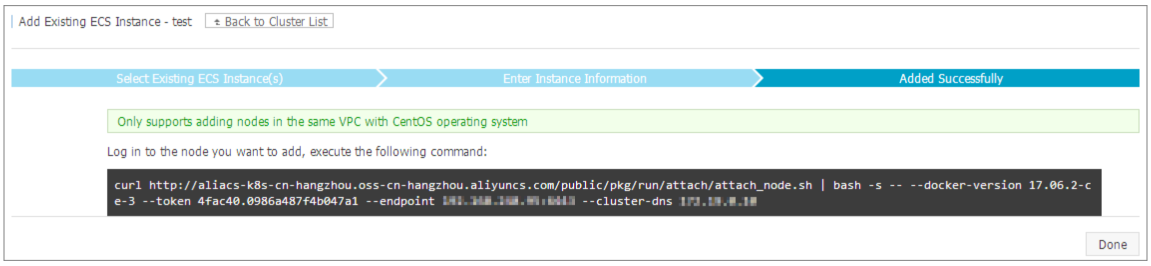
Cluster ID/Name : [redacted] / k8s-cluster

Information of the cluster to which to add the ECS instance(s).

Instance Information :

Instance ID	Instance Name
[redacted]	[redacted]7da6e71d6ae8d29

c) Copy the command.



Add Existing ECS Instance - test
Back to Cluster List
Select Existing ECS Instance(s)
Enter Instance Information
Added Successfully

Only supports adding nodes in the same VPC with CentOS operating system

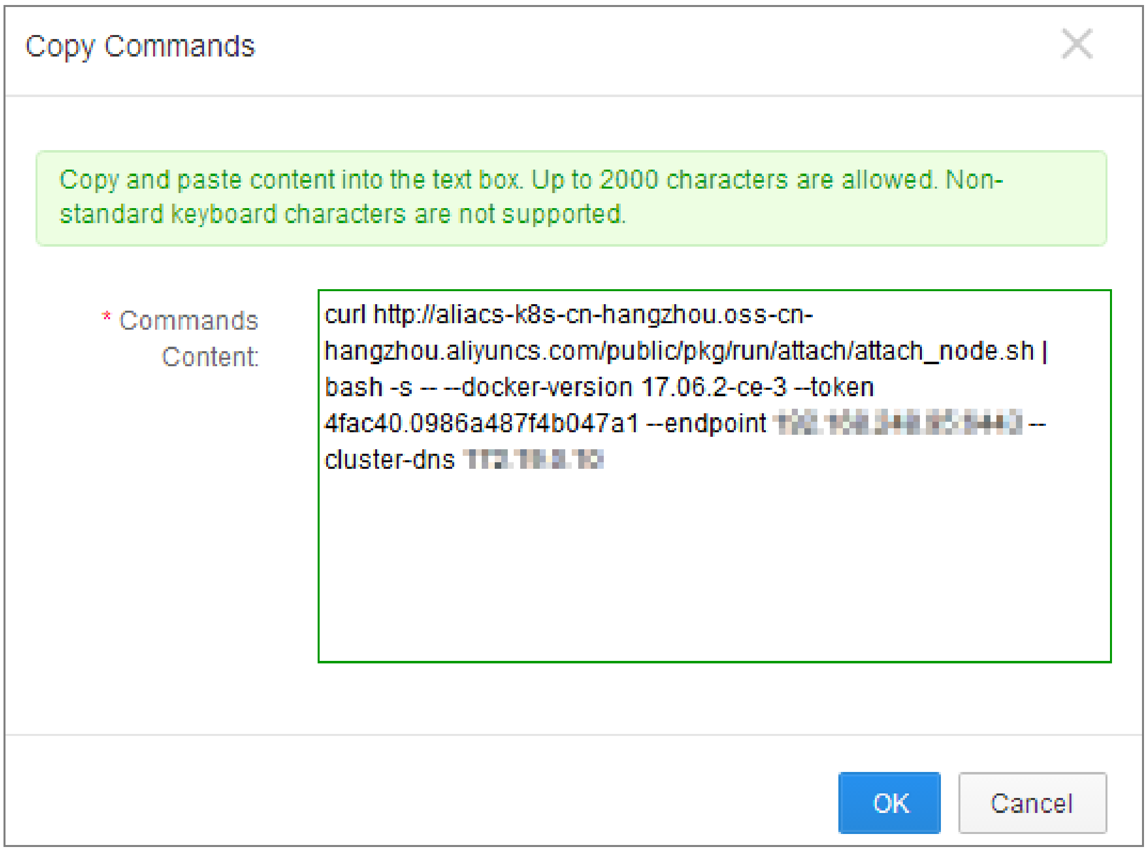
Log in to the node you want to add, execute the following command:

```
curl http://aliacs-k8s-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/public/pkg/run/attach/attach_node.sh | bash -s -- --docker-version 17.06.2-c
e-3 --token 4fac40.0986a487f4b047a1 --endpoint [redacted] --cluster-dns [redacted]
```

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.



Copy and paste content into the text box. Up to 2000 characters are allowed. Non-standard keyboard characters are not supported.

* Commands Content:

```
curl http://aliacs-k8s-cn-hangzhou.oss-cn-
hangzhou.aliyuncs.com/public/pkg/run/attach/attach_node.sh |
bash -s -- --docker-version 17.06.2-ce-3 --token
4fac40.0986a487f4b047a1 --endpoint [redacted] --
cluster-dns [redacted]
```

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.

1.4.2 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_48* 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
iz2ze2n6ep53tch701yh9zz Ready 19m v1.6.1-2+ed9e3d33a07093
```

```
iz2zeafr762wibijx39e5az Ready 7m v1.6.1-2+ed9e3d33a07093
iz2zeafr762wibijx39e5bz Ready 7m v1.6.1-2+ed9e3d33a07093
iz2zef4dnn9nos8elyr32kz Ready 14m v1.6.1-2+ed9e3d33a07093
iz2zeitvvo8enoreufstkmz Ready 11m v1.6.1-2+ed9e3d33a07093
```

View node list in Container Service console

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 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.

1.4.3 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 *Introduction to Host monitoring*.

Kubernetes clusters can now monitor resources by using application groups. For more information, see *#unique_51*.

1.4.4 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_53*.

Prerequisite

You have successfully created a Kubernetes cluster. For more information, see *#unique_31*.

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`.

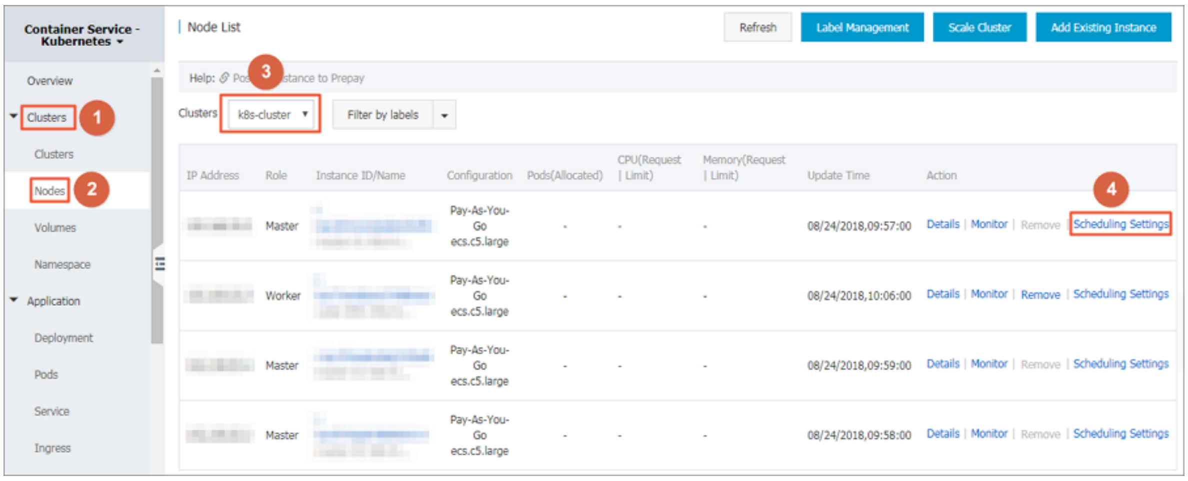
   Click Confirm in the displayed dialog box. The node label is deleted.

1.4.5 Set node scheduling

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

Procedure

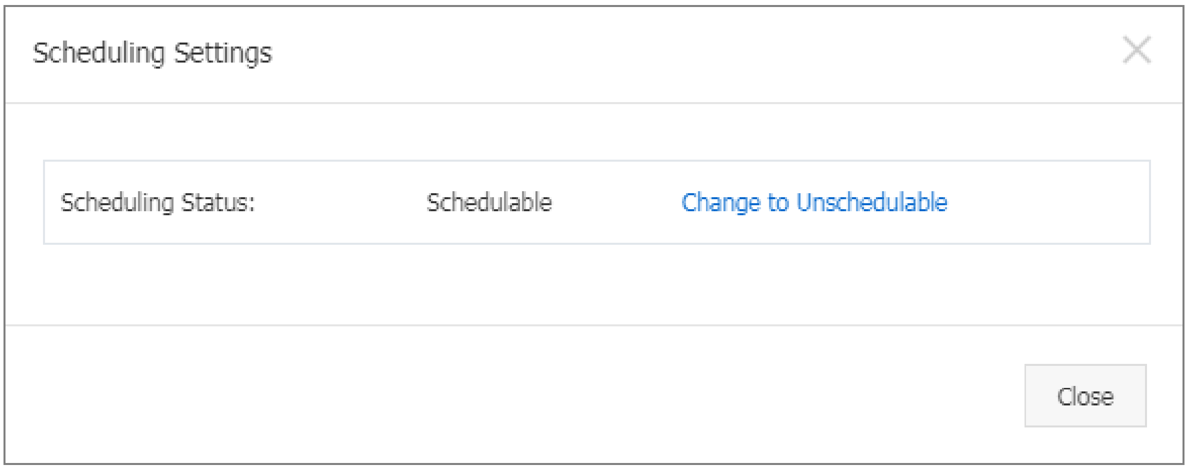
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** > **Nodes** to enter the Node List page.
3. Select a cluster, select a node under the cluster, and click **Schedule Settings** on the right.



IP Address	Role	Instance ID/Name	Configuration	Pods(Allocated)	CPU(Request | Limit)	Memory(Request | Limit)	Update Time	Action
[redacted]	Master	[redacted]	Pay-As-You-Go ecs.c5.large	-	-	-	08/24/2018,09:57:00	Details | Monitor | Remove | Scheduling Settings
[redacted]	Worker	[redacted]	Pay-As-You-Go ecs.c5.large	-	-	-	08/24/2018,10:06:00	Details | Monitor | Remove | Scheduling Settings
[redacted]	Master	[redacted]	Pay-As-You-Go ecs.c5.large	-	-	-	08/24/2018,09:59:00	Details | Monitor | Remove | Scheduling Settings
[redacted]	Master	[redacted]	Pay-As-You-Go ecs.c5.large	-	-	-	08/24/2018,09:58:00	Details | Monitor | Remove | Scheduling Settings

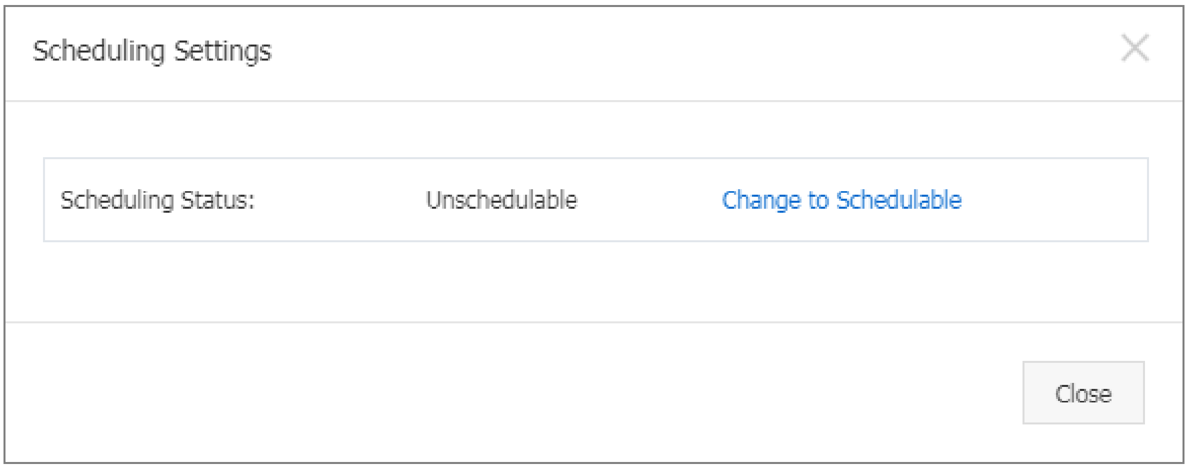
4. Set node scheduling in the displayed dialog box. In this example, click **Change to Unschedulable** to set the node to unschedulable.

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.


Scheduling Settings

Scheduling Status: Schedulable Change to Unschedulable

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


Scheduling Settings

Scheduling Status: Unschedulable Change to Schedulable

What's next

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

1.4.6 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

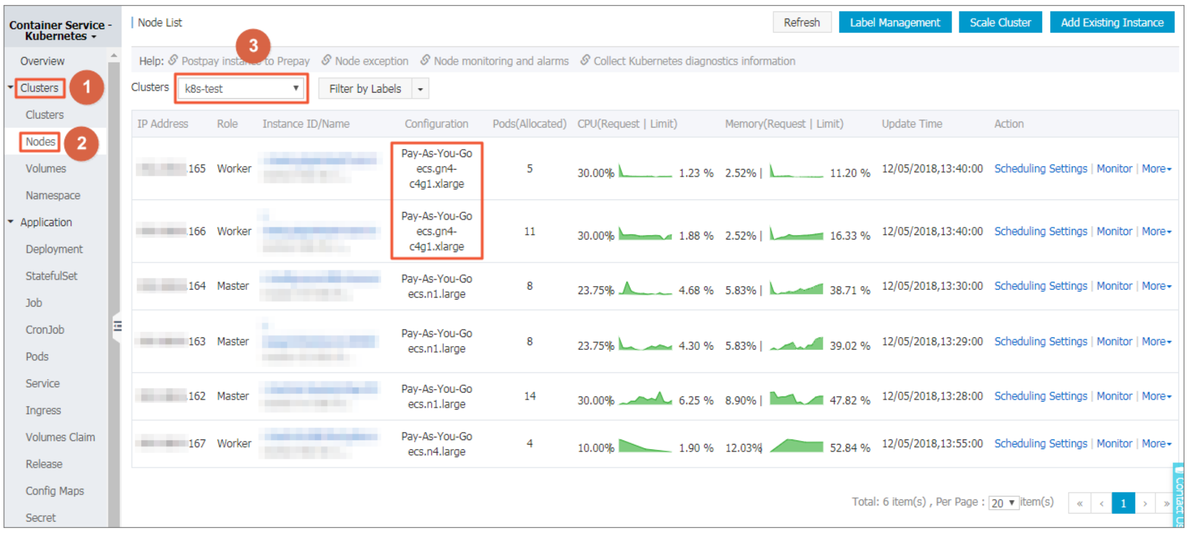
When deploying NVIDIA GPU nodes, Kubernetes that runs on Alibaba Cloud discovers the GPU attribute and exposes it as the node label information. Node labels provide the following benefits:

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 Kubernetes, choose **Clusters** > **Nodes**.

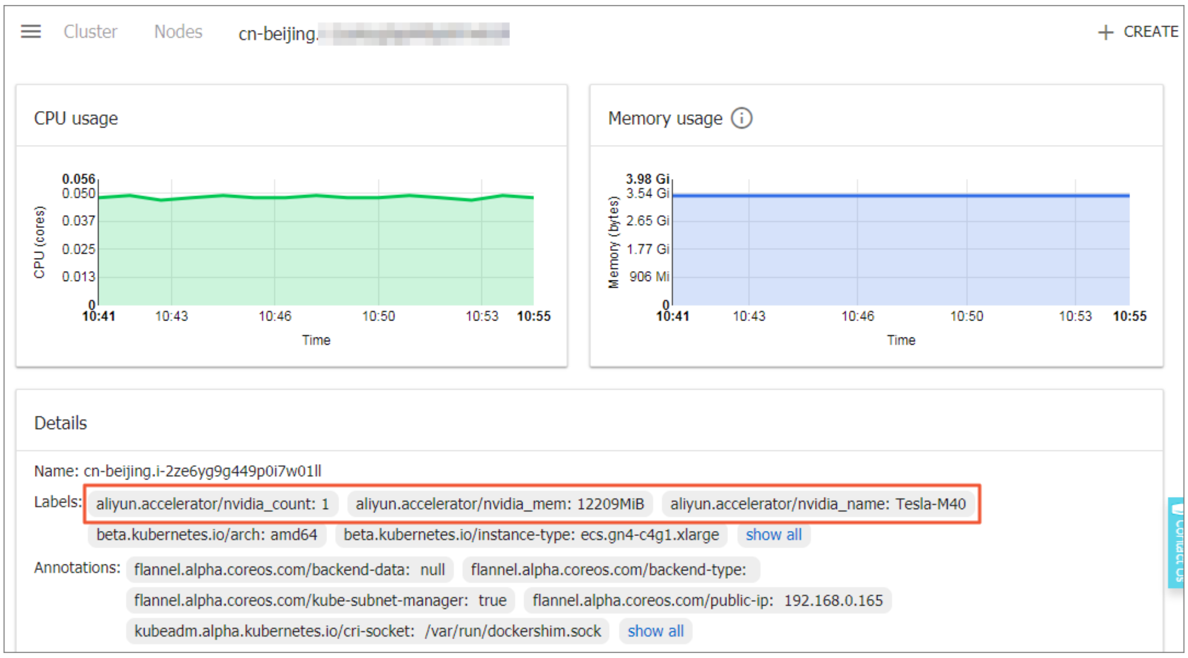
Note:
In this example, the cluster has three Worker nodes of which two Worker nodes are mounted with GPU devices. You need to view the node IP addresses for verification.



IP Address	Role	Instance ID/Name	Configuration	Pods(Allocated)	CPU(Request | Limit)	Memory(Request | Limit)	Update Time	Action
[redacted].165	Worker	[redacted]	Pay-As-You-Go ecs.gn4-c4g1.xlarge	5	30.00% | 1.23 %	2.52% | 11.20 %	12/05/2018,13:40:00	Scheduling Settings | Monitor | More
[redacted].166	Worker	[redacted]	Pay-As-You-Go ecs.gn4-c4g1.xlarge	11	30.00% | 1.88 %	2.52% | 16.33 %	12/05/2018,13:40:00	Scheduling Settings | Monitor | More
[redacted].164	Master	[redacted]	Pay-As-You-Go ecs.n1.large	8	23.75% | 4.68 %	5.83% | 38.71 %	12/05/2018,13:30:00	Scheduling Settings | Monitor | More
[redacted].163	Master	[redacted]	Pay-As-You-Go ecs.n1.large	8	23.75% | 4.30 %	5.83% | 39.02 %	12/05/2018,13:29:00	Scheduling Settings | Monitor | More
[redacted].162	Master	[redacted]	Pay-As-You-Go ecs.n1.large	14	30.00% | 6.25 %	8.90% | 47.82 %	12/05/2018,13:28:00	Scheduling Settings | Monitor | More
[redacted].167	Worker	[redacted]	Pay-As-You-Go ecs.n4.large	4	10.00% | 1.90 %	12.03% | 52.84 %	12/05/2018,13:55:00	Scheduling Settings | Monitor | More

Total: 6 item(s) , Per Page : 20 item(s) « ‹ 1 › »

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.



Details

Name: cn-beijing.i-2ze6yg9g449p0i7w01ll

Labels: aliyun.accelerator/nvidia_count: 1 aliyun.accelerator/nvidia_mem: 12209MiB aliyun.accelerator/nvidia_name: Tesla-M40 beta.kubernetes.io/arch: amd64 beta.kubernetes.io/instance-type: ecs.gn4-c4g1.xlarge show all

Annotations: flannel.alpha.coreos.com/backend-data: null flannel.alpha.coreos.com/backend-type: flannel.alpha.coreos.com/kube-subnet-manager: true flannel.alpha.coreos.com/public-ip: 192.168.0.165 kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock show all

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-2ze2dy2h9w97v65uuaft       Ready     master    2d         v1
.11.2
cn-beijing.i-2ze8o1a45qdv5q8a7luz       Ready     <none>    2d         v1
.11.2             #Compare this node with the node displayed in the
console to determine the GPU node.
cn-beijing.i-2ze8o1a45qdv5q8a7lv0       Ready     <none>    2d         v1
.11.2
cn-beijing.i-2ze9xylyn11vop7g5bwe       Ready     master    2d         v1
.11.2
cn-beijing.i-2zed5sw8snjniq6mf5e5       Ready     master    2d         v1
.11.2
cn-beijing.i-2zej9s0zijykp9pwf7lu       Ready     <none>    2d         v1
.11.2
```

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

```
# kubectl describe node cn-beijing.i-2ze8o1a45qdv5q8a7luz
Name:                   cn-beijing.i-2ze8o1a45qdv5q8a7luz
Roles:                  <none>
Labels:                 aliyun.accelerator/nvidia_count=1
           #This field is important.
                        aliyun.accelerator/nvidia_mem=12209MiB
                        aliyun.accelerator/nvidia_name=Tesla-M40
                        beta.kubernetes.io/arch=amd64
                        beta.kubernetes.io/instance-type=ecs.gn4-c4g1.
xlarge
                        beta.kubernetes.io/os=linux
                        failure-domain.beta.kubernetes.io/region=cn-
beijing
```

```
                    failure-domain.beta.kubernetes.io/zone=cn-
beijing-a
                    kubernetes.io/hostname=cn-beijing.i-2ze8o1a45q
dv5q8a7luz
 ......
```

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

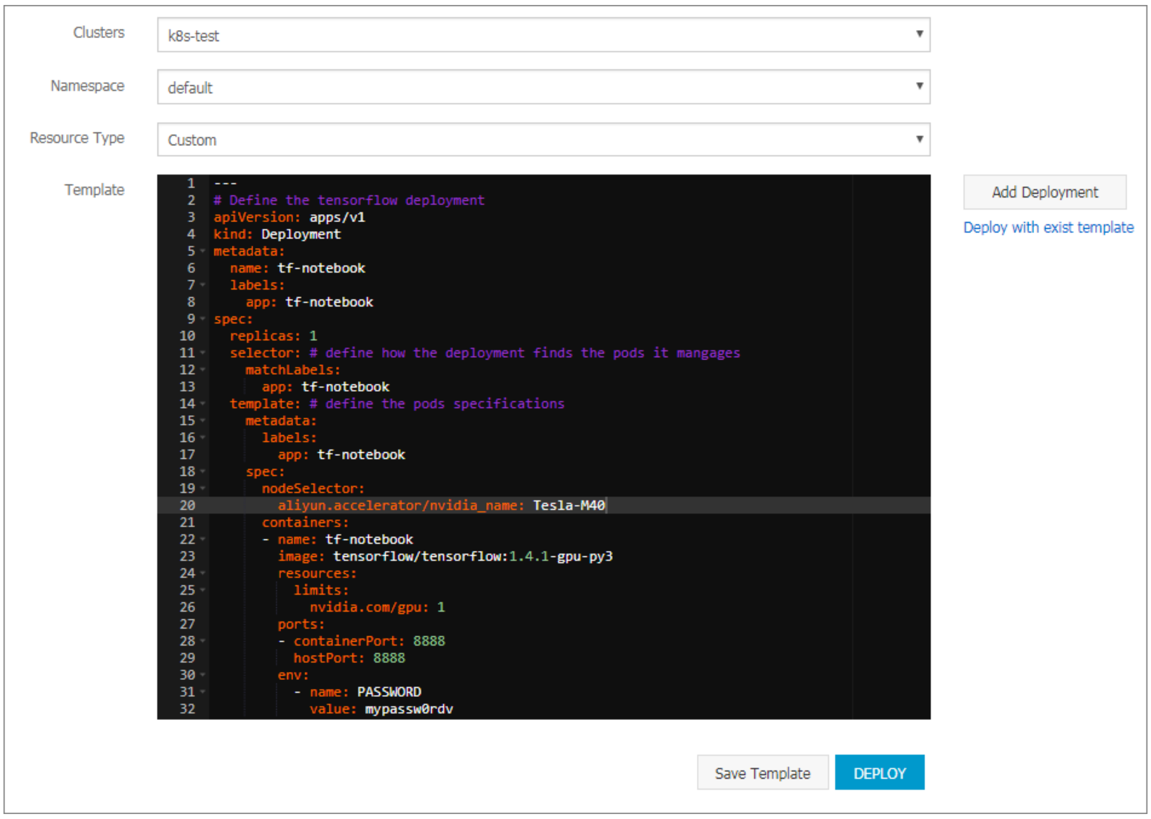
Key	Value
`aliyun.accelerator/nvidia_count`	Number of GPU cores
`aliyun.accelerator/nvidia_mem`	GPU memory in MiB
`aliyun.accelerator/nvidia_name`	Name of the GPU computing card of the NVIDIA device

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.accelerator/nvidia_name=Tesla-M40
NAME                                STATUS    ROLES     AGE
VERSION
cn-beijing.i-2ze8o1a45qdv5q8a7luz   Ready     <none>    2d        v1
.11.2
cn-beijing.i-2ze8o1a45qdv5q8a7lv0   Ready     <none>    2d        v1
.11.2
```

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

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



```
---
# Define the tensorflow deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tf-notebook
  labels:
    app: tf-notebook
spec:
  replicas: 1
  selector: # define how the deployment finds the pods it mangages
    matchLabels:
      app: tf-notebook
  template: # define the pods specifications
    metadata:
      labels:
        app: tf-notebook
    spec:
      nodeSelector:
        aliyun.accelerator/nvidia_name: Tesla-M40
      containers:
      - name: tf-notebook
        image: tensorflow/tensorflow:1.4.1-gpu-py3
        resources:
          limits:
            nvidia.com/gpu: 1
        ports:
        - containerPort: 8888
          hostPort: 8888
        env:
          - name: PASSWORD
            value: mypassw0rdv
```

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
    matchLabels:
      app: tf-notebook
  template: #Define the pod specifications.
    metadata:
      labels:
        app: tf-notebook
    spec:
      nodeSelector:
   #This field is important.
        aliyun.accelerator/nvidia_name: 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:
        - containerPort: 8888
```

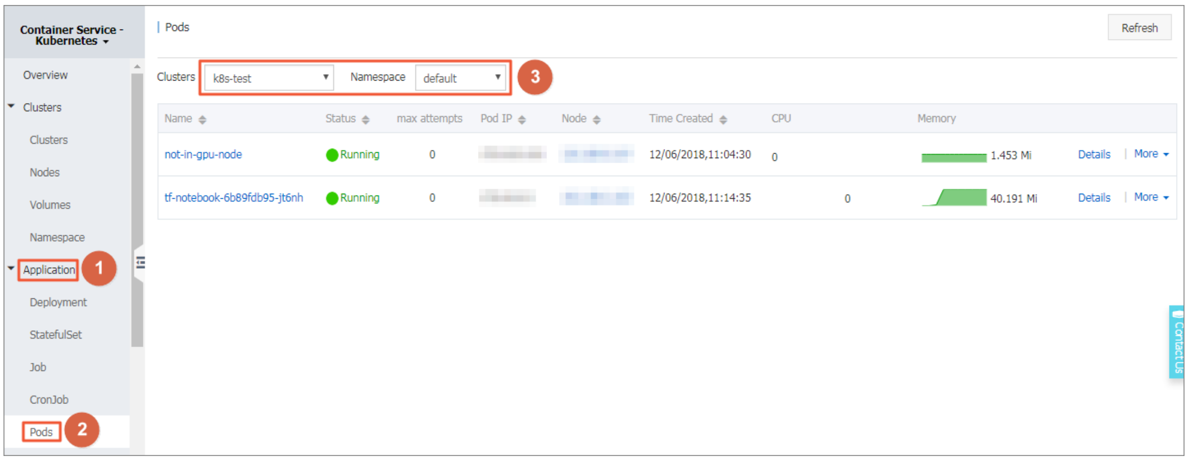
```
     hostPort: 8888
    env:
     - name: PASSWORD
      value: mypassw0rdv
```

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:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: aliyun.accelerator/nvidia_name
            operator: DoesNotExist
  containers:
  - name: not-in-gpu-node
    image: nginx
```

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



Clusters: k8s-test   Namespace: default

Name	Status	max attempts	Pod IP	Node	Time Created	CPU	Memory		
not-in-gpu-node	Running	0	[redacted]	[redacted]	12/06/2018,11:04:30	0	1.453 Mi	Details	More
tf-notebook-6b89fdb95-jt6nh	Running	0	[redacted]	[redacted]	12/06/2018,11:14:35	0	40.191 Mi	Details	More

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.

1.4.7 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

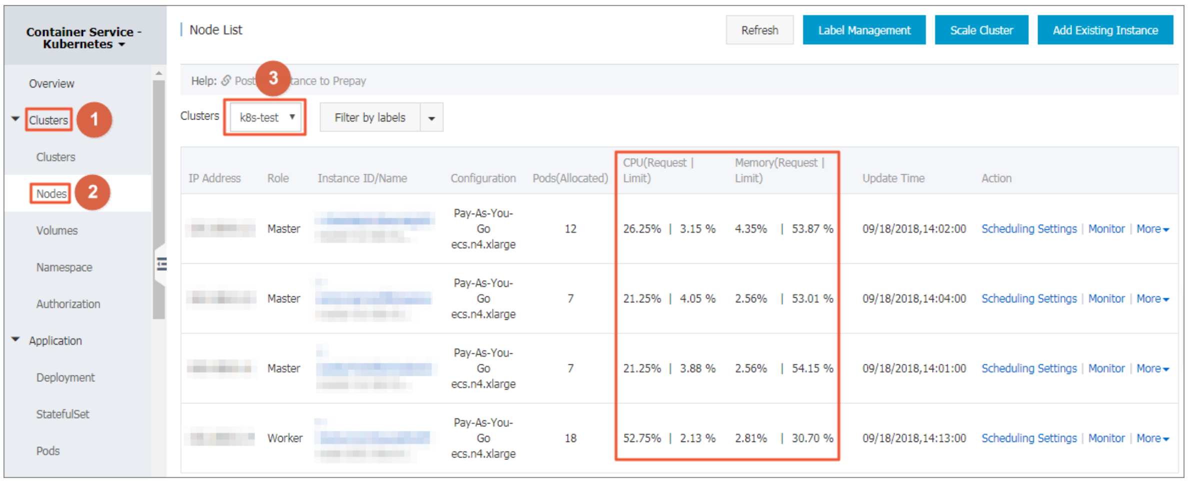
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** > **Nodes**.

   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.



IP Address	Role	Instance ID/Name	Configuration	Pods(Allocated)	CPU(Request | Limit)	Memory(Request | Limit)	Update Time	Action
[redacted]	Master	[redacted]	Pay-As-You-Go ecs.n4.xlarge	12	26.25% | 3.15 %	4.35% | 53.87 %	09/18/2018,14:02:00	Scheduling Settings | Monitor | More
[redacted]	Master	[redacted]	Pay-As-You-Go ecs.n4.xlarge	7	21.25% | 4.05 %	2.56% | 53.01 %	09/18/2018,14:04:00	Scheduling Settings | Monitor | More
[redacted]	Master	[redacted]	Pay-As-You-Go ecs.n4.xlarge	7	21.25% | 3.88 %	2.56% | 54.15 %	09/18/2018,14:01:00	Scheduling Settings | Monitor | More
[redacted]	Worker	[redacted]	Pay-As-You-Go ecs.n4.xlarge	18	52.75% | 2.13 %	2.81% | 30.70 %	09/18/2018,14:13:00	Scheduling Settings | Monitor | More

1.5 Namespaces

1.5.1 Create a namespace

Prerequisites

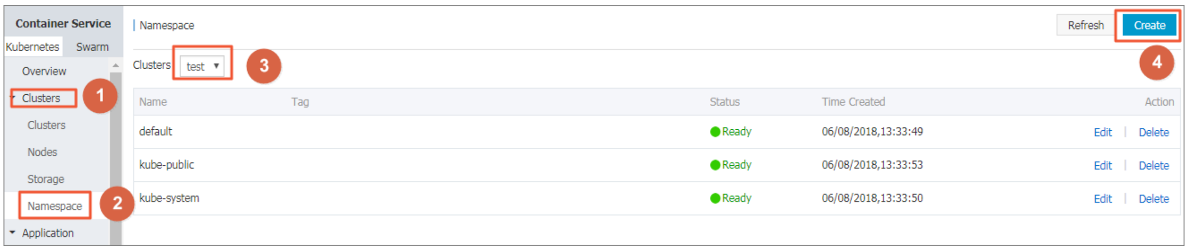
You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

Context

In Kubernetes clusters, you can use the Namespace function to create multiple virtual spaces. When the number of users in one cluster is large, multiple namespaces are used to divide the workspaces effectively and the cluster resources into different purposes. The namespace resources are assigned by using the *resource-quotas*.

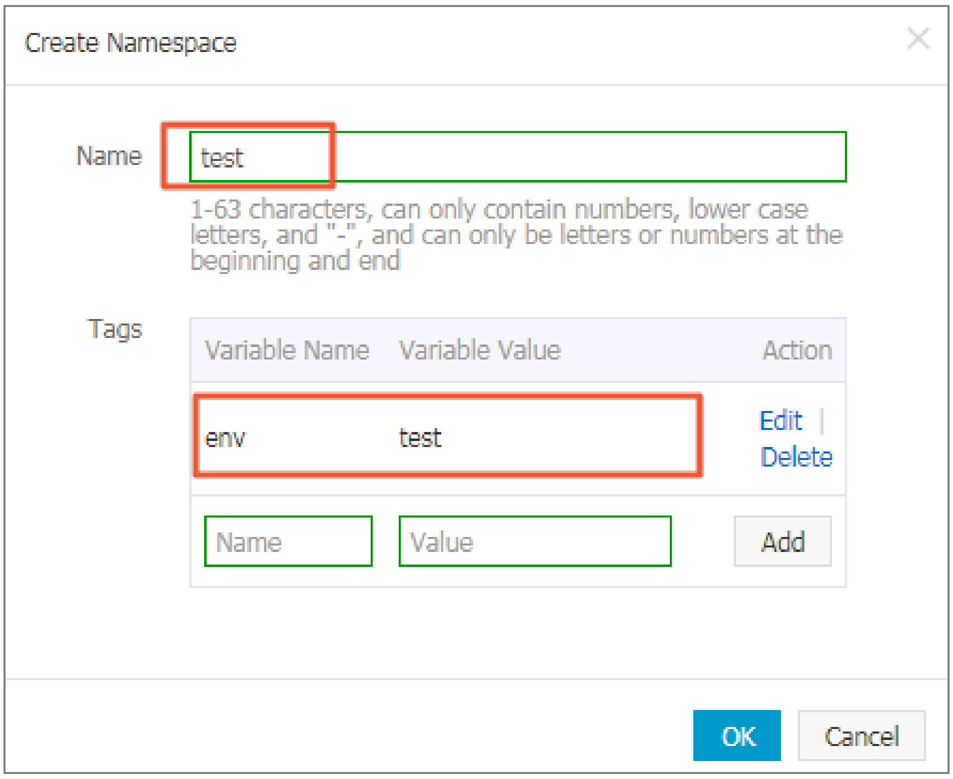
Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** > **Namespace** in the left-side navigation pane.
3. Select the cluster from the Clusters drop-down list and then click **Create** in the upper-right corner.



Name	Tag	Status	Time Created	Action
default		● Ready	06/08/2018,13:33:49	Edit | Delete
kube-public		● Ready	06/08/2018,13:33:53	Edit | Delete
kube-system		● Ready	06/08/2018,13:33:50	Edit | Delete

4. Configure the namespace in the displayed dialog box.



Name	test	
	1-63 characters, can only contain numbers, lower case letters, and "-", and can only be letters or numbers at the beginning and end	

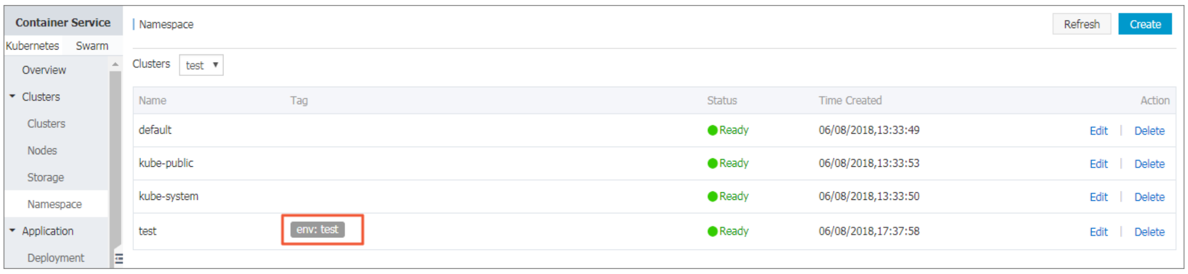
Tags

Variable Name	Variable Value	Action
env	test	Edit | Delete
Name	Value	Add

- **Name**: Enter the namespace name, which is 1–63 characters long, can only contain numbers, letters, and hyphens (-), and must start and end with a letter or number. In this example, enter test as the name.
- **Tags**: Add one or more tags for the namespace to identify the characteristics of the namespace, for example, to identify this namespace to be used for the test environment.

  You can enter the variable name and variable value, and then click **Add** on the right to add a tag for the namespace.

5. Click **OK** after completing the configurations.
6. The namespace test is successfully created and displayed in the namespace list.



Clusters test

Name	Tag	Status	Time Created	Action
default		● Ready	06/08/2018,13:33:49	Edit | Delete
kube-public		● Ready	06/08/2018,13:33:53	Edit | Delete
kube-system		● Ready	06/08/2018,13:33:50	Edit | Delete
test	env: test	● Ready	06/08/2018,17:37:58	Edit | Delete

1.5.2 Configure resource quotas and limits for namespaces

You can configure resource quotas and limits for namespaces through the Container Services console.

Prerequisites

- You have created an Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a namespace **test**. For more information, see *Create a namespace*.
- You have connected to the master node address of the cluster. For more information, see *Connect to a Kubernetes cluster by using kubectl*.

Context

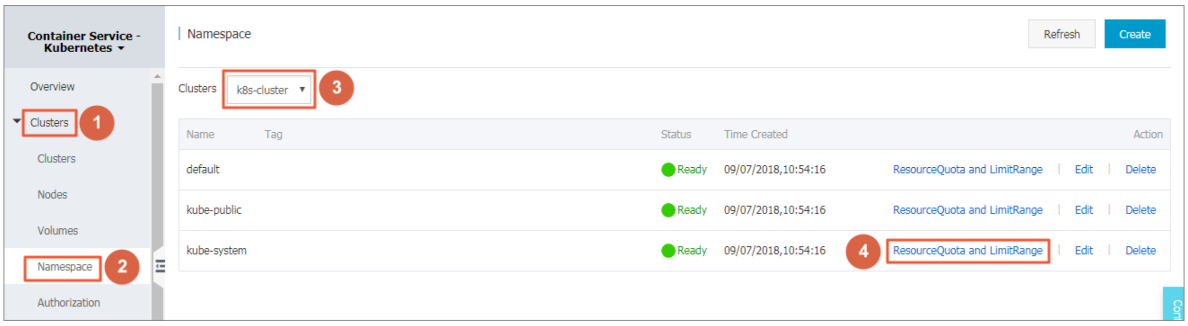
By default, a running pod can use the CPU and memory of nodes unlimitedly, which means any pod can use the computing resources of the cluster unlimitedly, and the pods of a namespace may use up the cluster resources.

One of the important functions of namespaces is to act as a virtual cluster for multiple purposes and meeting the requirements of multiple users. Therefore, configuring the resource quotas for a namespace is a kind of best practice.

You can configure the resource quotas for a namespace, including CPU, memory, and number of pods. For more information, see *Resource Quotas*.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** > **Namespace**. Select a cluster and click **ResourceQuota and LimitRange** on the right of the namespace **test**.



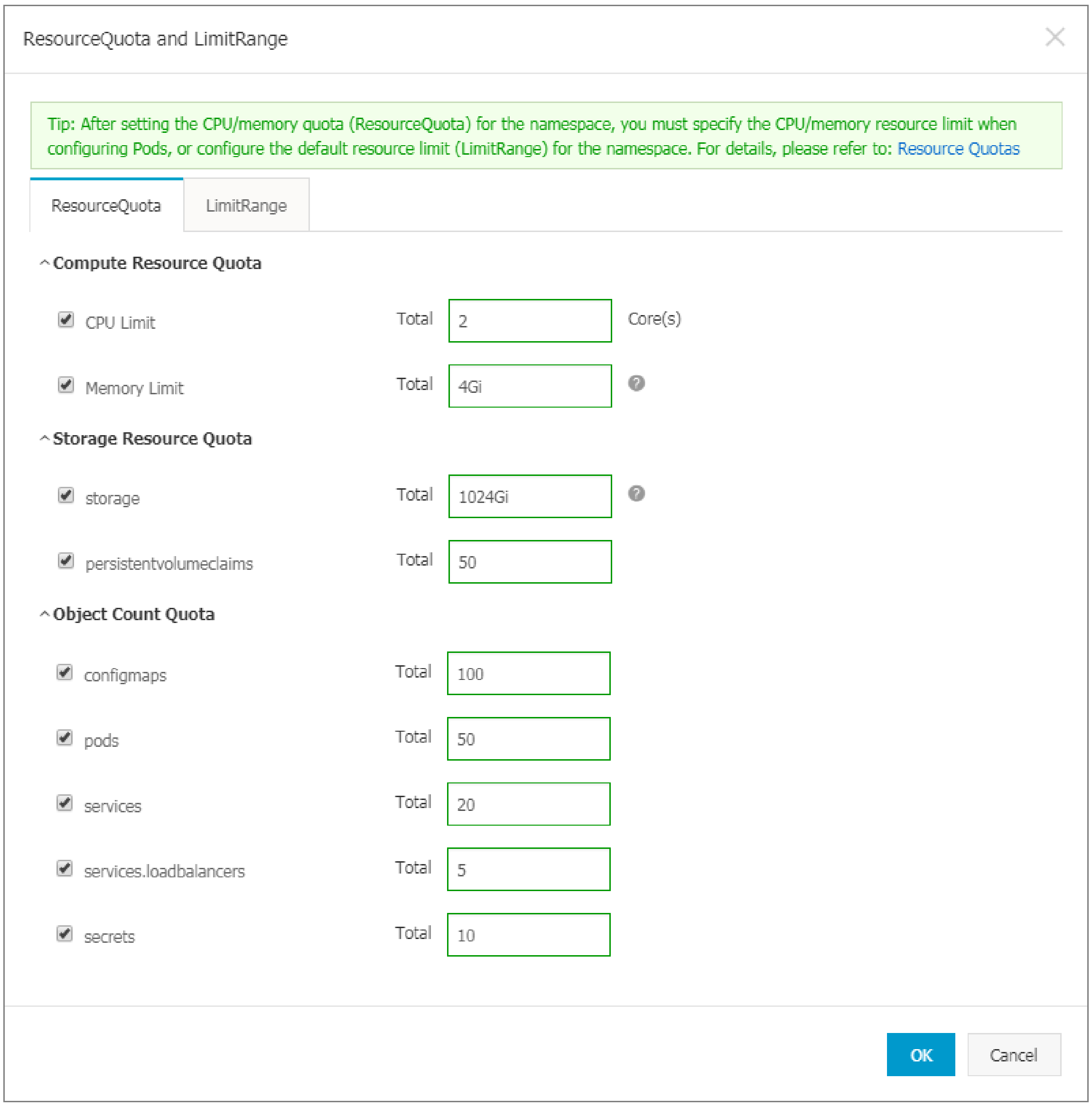
Name	Tag	Status	Time Created	Action
default		Ready	09/07/2018,10:54:16	ResourceQuota and LimitRange | Edit | Delete
kube-public		Ready	09/07/2018,10:54:16	ResourceQuota and LimitRange | Edit | Delete
kube-system		Ready	09/07/2018,10:54:16	ResourceQuota and LimitRange | Edit | Delete

3. On the ResourceQuota and LimitRange dialog box, you can quickly set resource quotas and default resource limits.

Note:

After setting the CPU/memory quota (ResourceQuota) for the namespace, you must specify the CPU/memory resource limit when configuring Pods, or configure the default resource limit (LimitRange) for the namespace. For details, please see *Resource Quotas*.

a) You can configure resource quotas for the namespace.



ResourceQuota and LimitRange

Tip: After setting the CPU/memory quota (ResourceQuota) for the namespace, you must specify the CPU/memory resource limit when configuring Pods, or configure the default resource limit (LimitRange) for the namespace. For details, please refer to: Resource Quotas

ResourceQuota | LimitRange

Compute Resource Quota

✓ CPU Limit	Total	2	Core(s)
✓ Memory Limit	Total	4Gi	

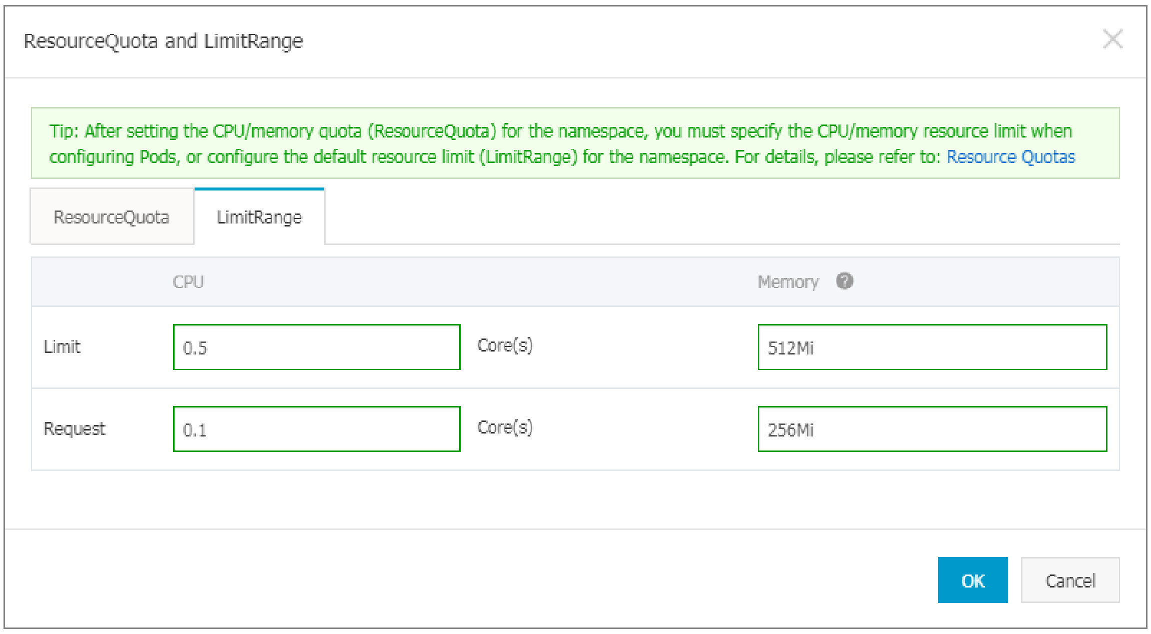
Storage Resource Quota

✓ storage	Total	1024Gi
✓ persistentvolumeclaims	Total	50

Object Count Quota

✓ configmaps	Total	100
✓ pods	Total	50
✓ services	Total	20
✓ services.loadbalancers	Total	5
✓ secrets	Total	10

b) You can control the container overhead by setting resource limits and resource requests for containers under this namespace. For more information, see *https://kubernetes.io//memory-default-namespace/*.


ResourceQuota and LimitRange

Tip: After setting the CPU/memory quota (ResourceQuota) for the namespace, you must specify the CPU/memory resource limit when configuring Pods, or configure the default resource limit (LimitRange) for the namespace. For details, please refer to: Resource Quotas

	CPU		Memory
Limit	0.5	Core(s)	512Mi
Request	0.1	Core(s)	256Mi

4. You have configured resource quotas and limits for the namespace. Connect to the master node connection address and execute the following commands to view the resources of the namespace:

```
#kubectl get limitrange,ResourceQuota -n test
NAME AGE
limitrange/limits 8m

NAME AGE
resourcequota/quota 8m


# kubectl describe limitrange/limits resourcequota/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
persistentvolumeclaims 0 50
pods 0 50
requests.storage 0 1Ti
secrets 1 10
services 0 20
```

```
services.loadbalancers 0 5
```

1.5.3 Update a namespace

Prerequisites

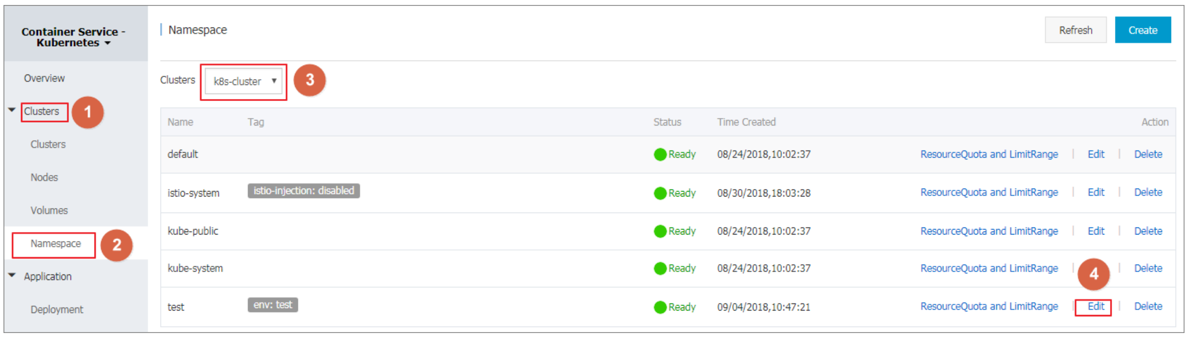
- You have created an Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a namespace **test**. For more information, see *Create a namespace*.

Context

You can update a namespace to add, modify, or delete the namespace tags.

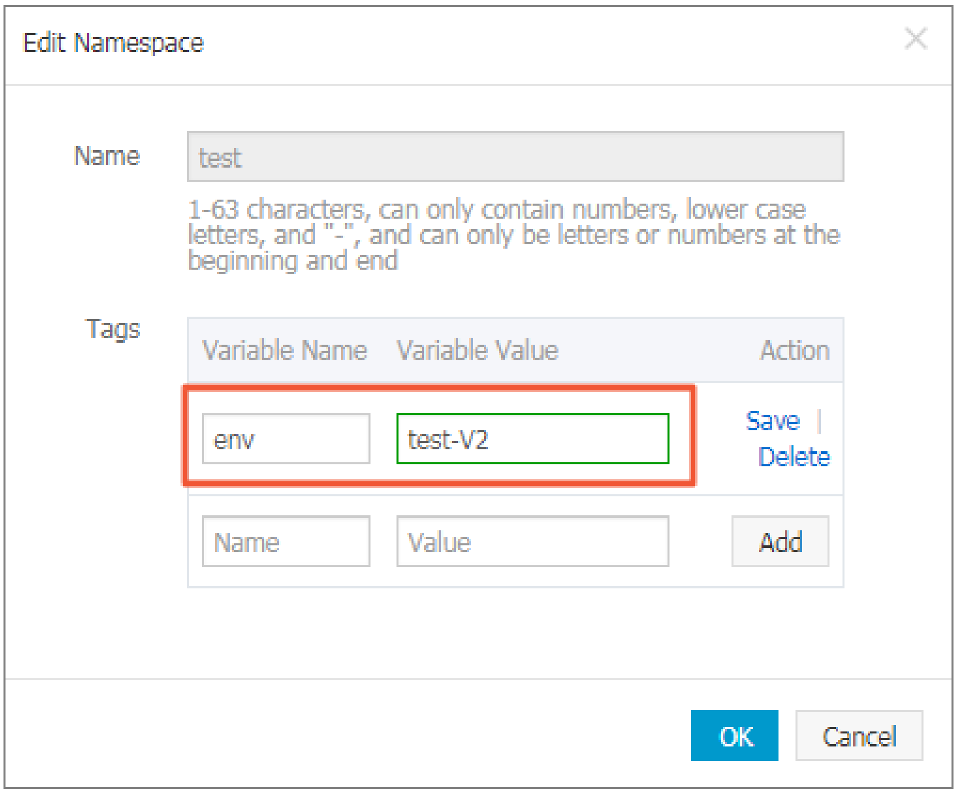
Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** > **Namespace** in the left-side navigation pane.
3. Select the cluster from the Clusters drop-down list and click **Edit** at the right of the cluster.



Name	Tag	Status	Time Created	Action
default		Ready	08/24/2018,10:02:37	ResourceQuota and LimitRange | Edit | Delete
istio-system	istio-injection: disabled	Ready	08/30/2018,18:03:28	ResourceQuota and LimitRange | Edit | Delete
kube-public		Ready	08/24/2018,10:02:37	ResourceQuota and LimitRange | Edit | Delete
kube-system		Ready	08/24/2018,10:02:37	ResourceQuota and LimitRange | Edit | Delete
test	env: test	Ready	09/04/2018,10:47:21	ResourceQuota and LimitRange | Edit | Delete

4. In the displayed dialog box, click**Edit** to update the namespace tags. For example, change the tag to `env:test-V2` and click**Save**.


Edit Namespace

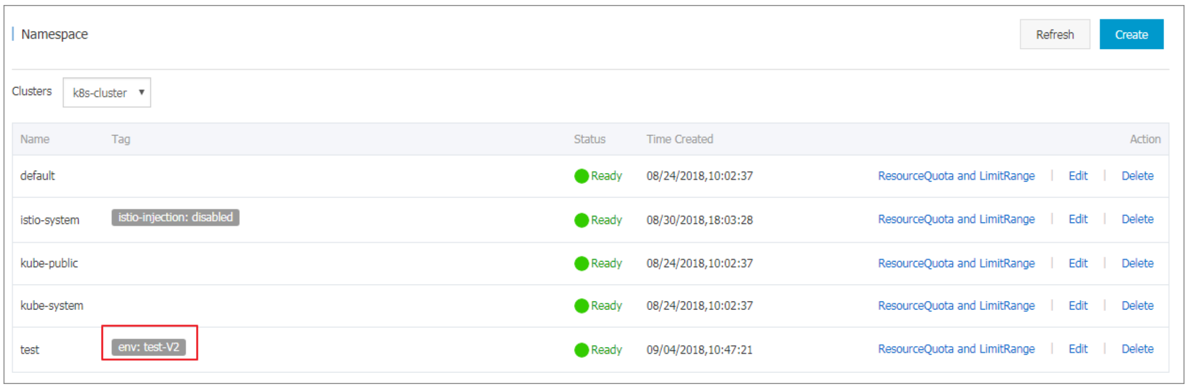
Name: test

1-63 characters, can only contain numbers, lower case letters, and "-", and can only be letters or numbers at the beginning and end

Tags

Variable Name	Variable Value	Action
env	test-V2	Save | Delete
Name	Value	Add

5. Click Save on the right and then click **OK**. The updated namespace tag is displayed in the namespace list.


Namespace

Clusters: k8s-cluster

Name	Tag	Status	Time Created	Action
default		Ready	08/24/2018,10:02:37	ResourceQuota and LimitRange | Edit | Delete
istio-system	istio-injection: disabled	Ready	08/30/2018,18:03:28	ResourceQuota and LimitRange | Edit | Delete
kube-public		Ready	08/24/2018,10:02:37	ResourceQuota and LimitRange | Edit | Delete
kube-system		Ready	08/24/2018,10:02:37	ResourceQuota and LimitRange | Edit | Delete
test	env: test-V2	Ready	09/04/2018,10:47:21	ResourceQuota and LimitRange | Edit | Delete

1.5.4 Delete a namespace

You can delete the namespaces that are no longer in use.

Prerequisites

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

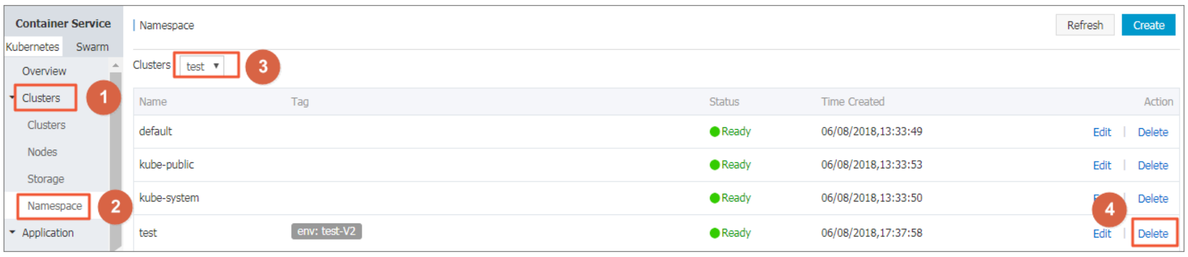
Context

Note:

Deleting a namespace also deletes all of its resource objects, so proceed with caution.

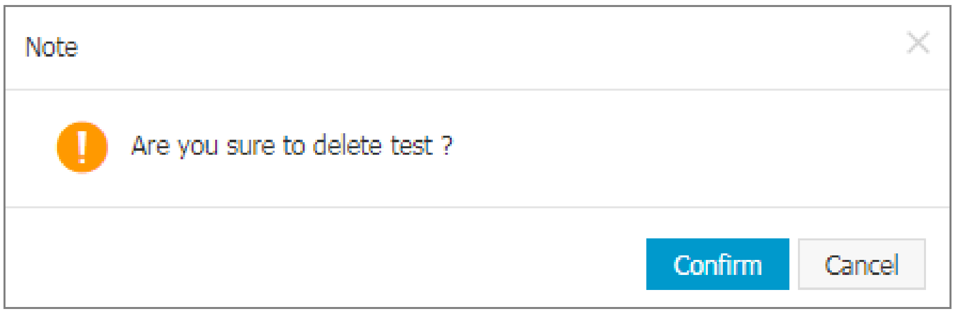
Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** > **Namespace** in the left-side navigation pane.
3. Select the cluster from the Clusters drop-down list and click **Delete** at the right of the cluster.



Name	Tag	Status	Time Created	Action
default		● Ready	06/08/2018,13:33:49	Edit | Delete
kube-public		● Ready	06/08/2018,13:33:53	Edit | Delete
kube-system		● Ready	06/08/2018,13:33:50	Delete
test	env: test-V2	● Ready	06/08/2018,17:37:58	Edit | Delete

4. Click **Confirm** in the displayed dialog box.



Are you sure to delete test ?

5. The namespace is deleted from the namespace list and its resource objects are also deleted.

1.6 Applications

1.6.1 Create a deployment application by using an image

You can use an image to create an Nginx application that is accessible for the Internet.

Prerequisites

Create a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

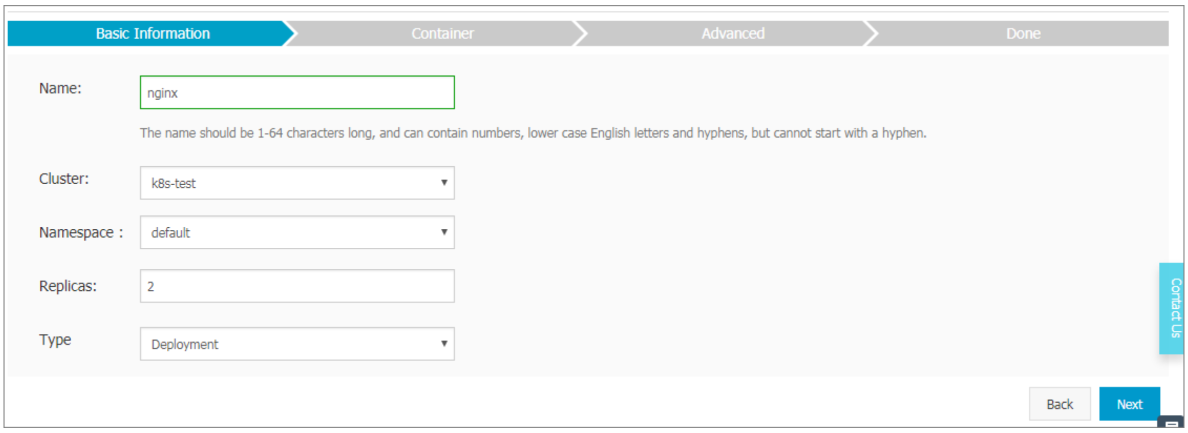
Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane, and 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.



Name: nginx

The name should be 1-64 characters long, and can contain numbers, lower case English letters and hyphens, but cannot start with a hyphen.

Cluster: k8s-test

Namespace : default

Replicas: 2

Type Deployment

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.

     Besides, you can 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 configured image is consistent with that of the cached image, Container Service reuses the cached image rather than pull the same image 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 an image when deploying an application so as to make sure the latest image and codes are 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/*.

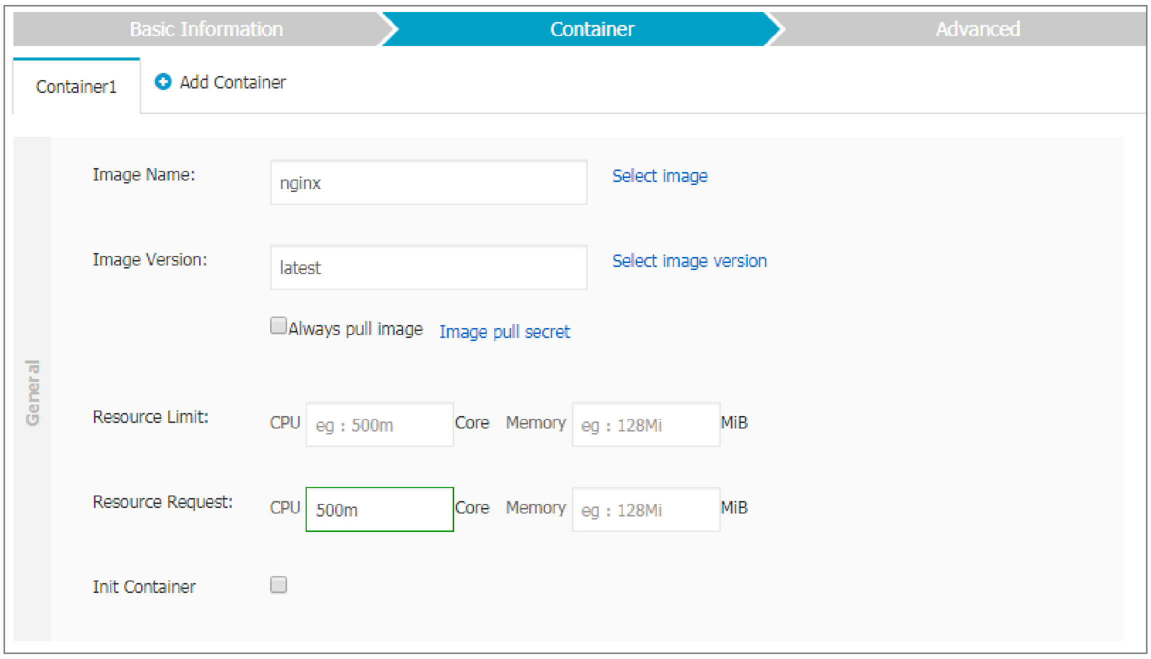


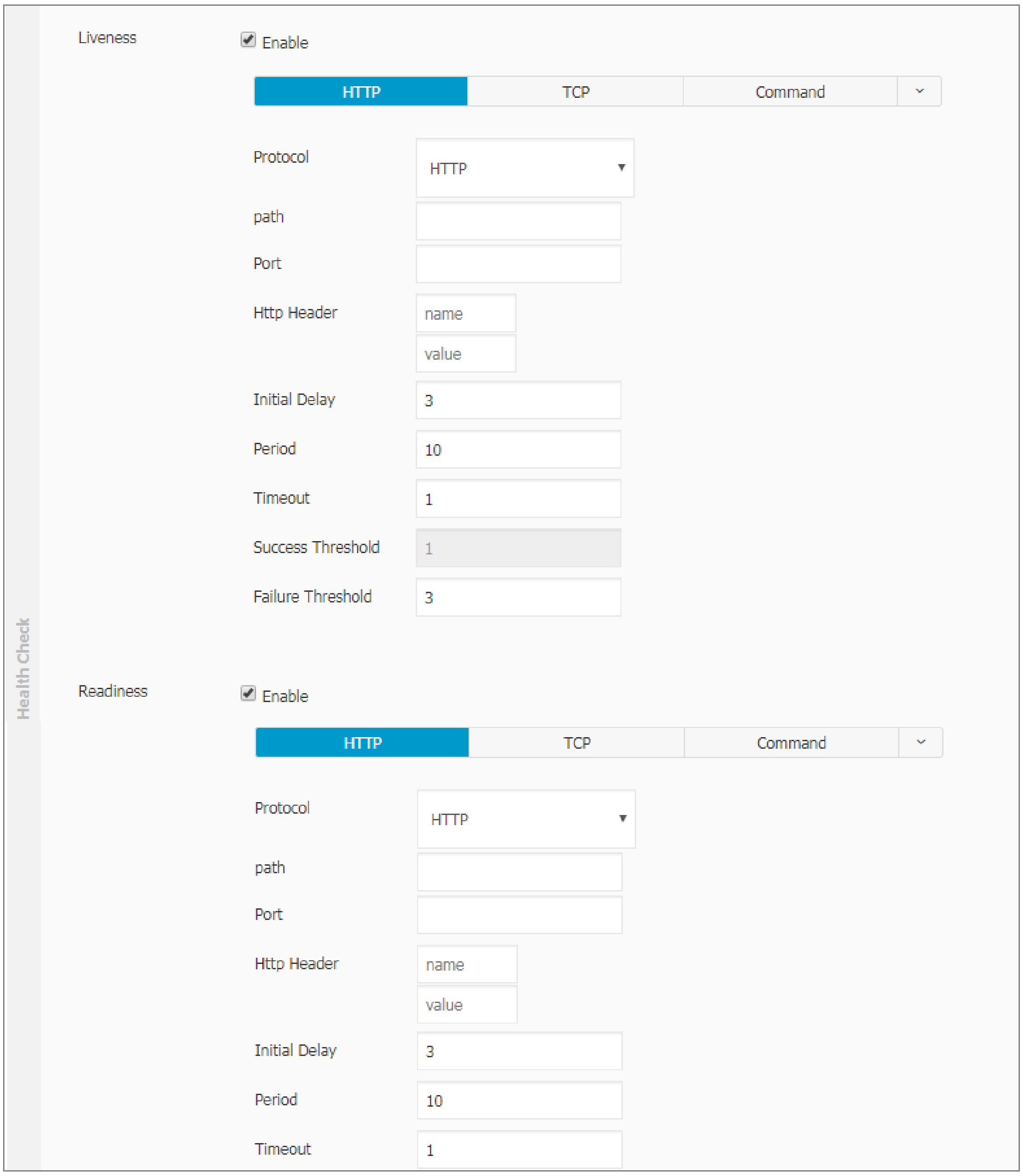
Image Name:	nginx
Image Version:	latest
	Always pull image
Resource Limit:	CPU eg : 500m Core Memory eg : 128Mi MiB
Resource Request:	CPU 500m Core Memory eg : 128Mi MiB
Init Container	☐

b) Optional: Configure environment variables.

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*.



Liveness ✓ Enable

Protocol	HTTP
path	
Port	
Http Header	name / value
Initial Delay	3
Period	10
Timeout	1
Success Threshold	1
Failure Threshold	3

Readiness ✓ Enable

Protocol	HTTP
path	
Port	
Http Header	name / value
Initial Delay	3
Period	10
Timeout	1

Request method	Configuration 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

Request method	Configuration description
	number must be in the range of 1 to 65535. • HTTP Header: Custom headers in the HTTP request. HTTP allows repeated headers. Supports key-value configurat ion. • 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.
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.

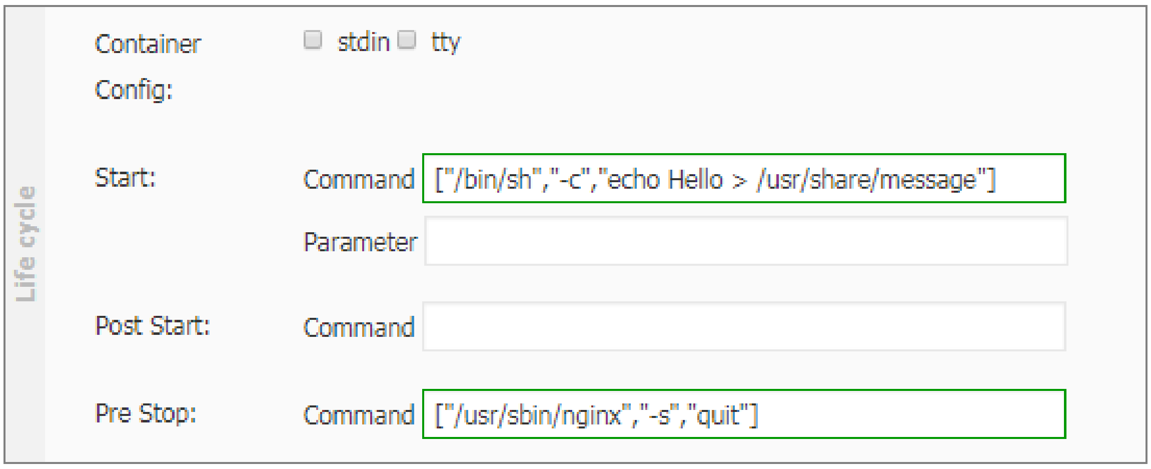
Request method	Configuration description
	• Initial Delay (in seconds): Namely, the **initialDelaySeconds**. Seconds for the first liveness or readiness probe has to wait after the container is started. The default is 15. • Period (in seconds): Namely, the **periodseconds**. Intervals at which the probe 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.
Command	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.

Request method	Configuration description
	• 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) Configure the lifecycle rule.

You can configure the following parameters for the container lifecycle: 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**: Select the stdin check box to enable standard input for the container. Select the tty check box to assign an virtual terminal to for the container to send signals to the container. These two options are usually used together, which indicates to bind the terminal (tty) to the container standard input (stdin). For example, an interactive program obtains standard input from you and then displays the obtained standard input in the terminal.
- **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.



Container Config: stdin tty

Start: Command ["/bin/sh","-c","echo Hello > /usr/share/message"]

Parameter

Post Start: Command

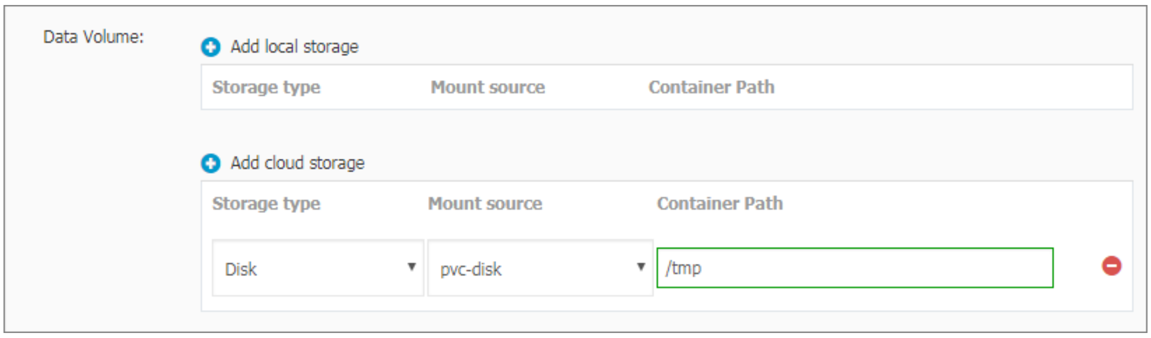
Pre Stop: Command ["/usr/sbin/nginx","-s","quit"]

e) Optional: 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 cloud disk as the data volume and mount the cloud disk to the `/tmp` container path. Then container data generated in this path are stored to the cloud disk.



Data Volume:

Add local storage

Storage type	Mount source	Container Path

Add cloud storage

Storage type	Mount source	Container Path
Disk	pvc-disk	/tmp

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

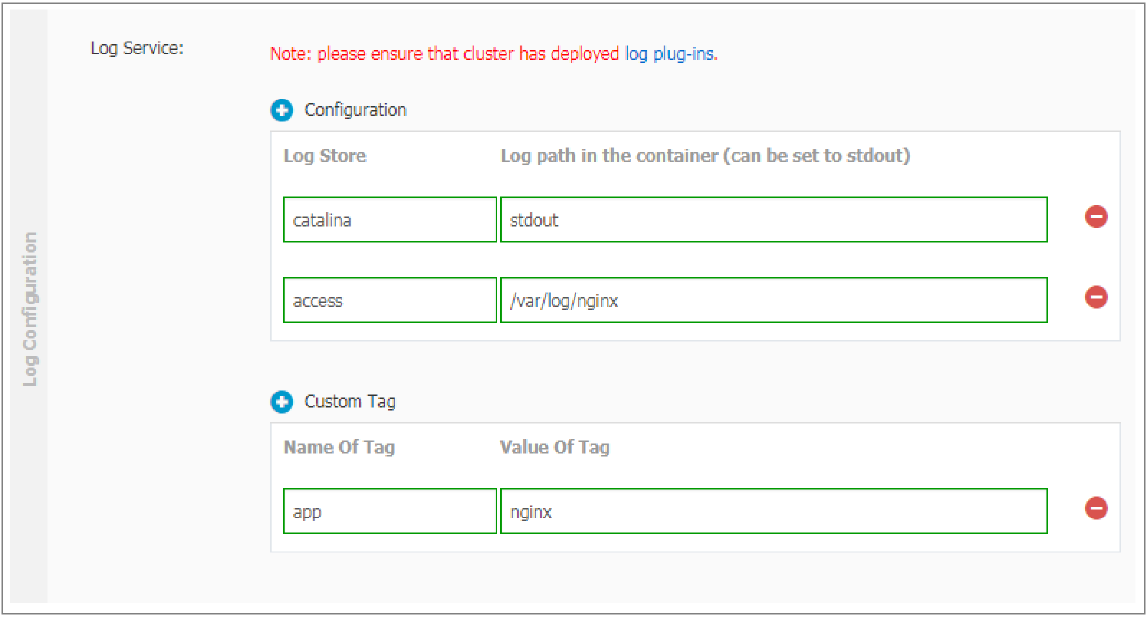
Note:

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 configured custom tags. The customized tags are collected to the container output logs. A custom tag can help you tag container logs, providing convenience to log analysis such as log statistics and filter.



Log Store	Log path in the container (can be set to stdout)
catalina	stdout
access	/var/log/nginx

Name Of Tag	Value Of Tag
app	nginx

5. Click **Next** after completing the configurations.
6. Configure advanced settings.

a) Configure **Access Control**.

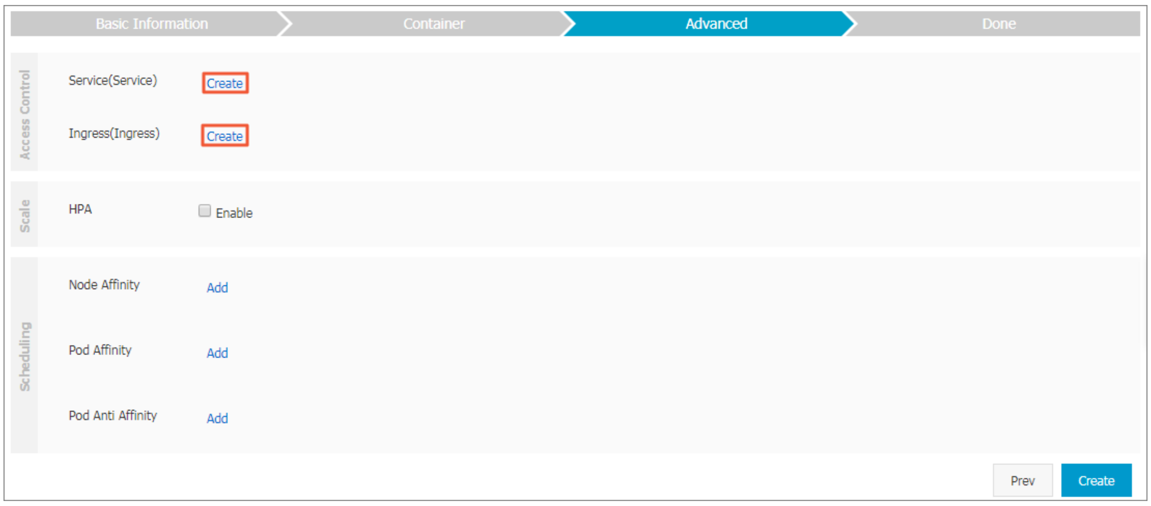
You can configure how to expose the backend pod and click **Create**. In this example, select Cluster IP Service and Ingress to create an nginx application that is accessible for Internet.

Note:

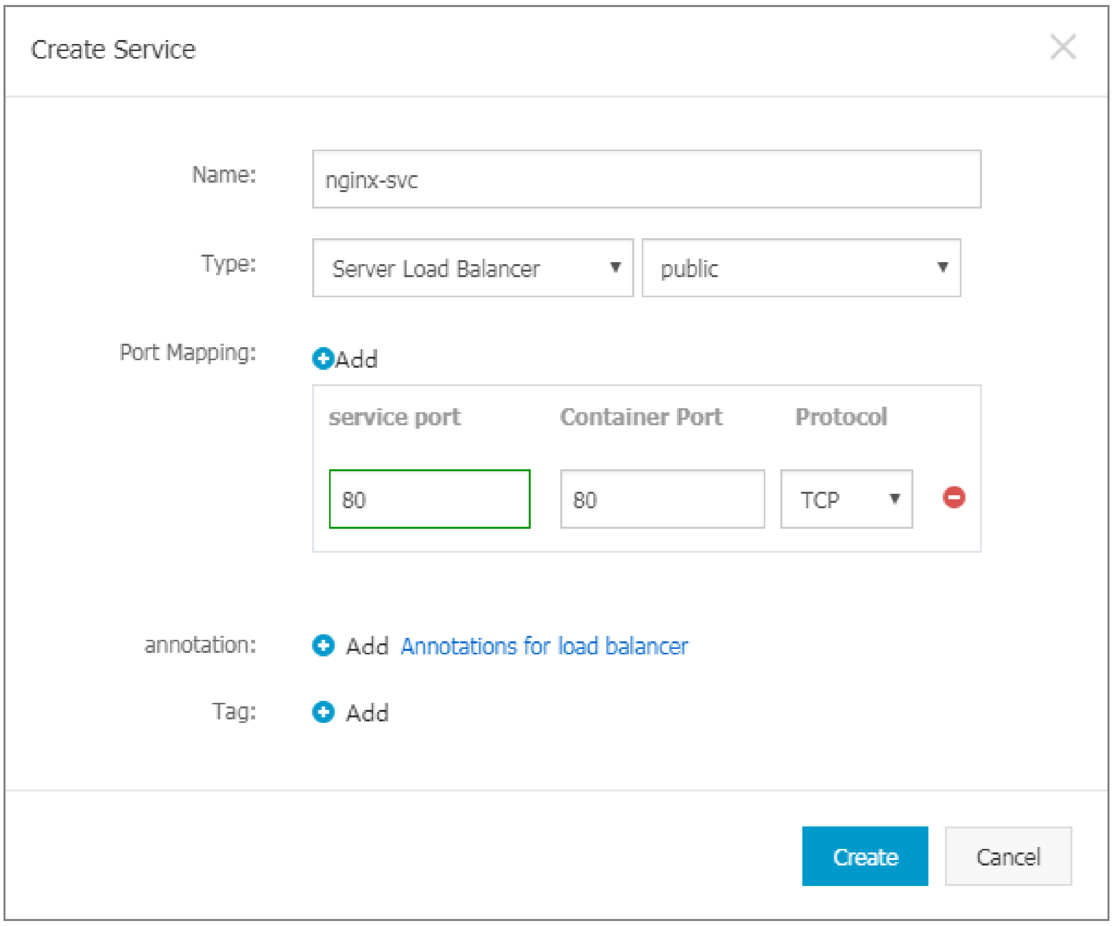
To meet communication requirements of the application, you can configure access control based on your needs:

- Internal applications: For applications that work only inside a cluster, you can create services of Cluster IP or Node Port for internal communication as needed.

- External applications: For applications that need to be exposed to Internet, you can configure access control by using one of the following methods:
  - Create a service of Server Load Balancer: Use the Server Load Balancer (SLB) service provided by Alibaba Cloud, which provides Internet accessibility for the application.
  - Create a service of ClusterIP or NodePort, and create Ingress: This method provides Internet accessibility through ingress. For more information, see *https://kubernetes.io/docs/concepts/services-networking/ingress/*.



1. Click **Create** at right of Service. Configure a service in the displayed dialog box, and then click **Create**.


Create Service

Name: nginx-svc

Type: Server Load Balancer public

Port Mapping: Add

service port	Container Port	Protocol
80	80	TCP

annotation: Add Annotations for load balancer

Tag: Add

- **Name**: You can enter your custom name. The default is `applicationname-svc`.
- **Type**: Select one from the following three service types.
  - ClusterIP: Expose the service by using the internal IP address of your cluster. With this type selected, the service is accessible only within the cluster.
  - NodePort: Expose the service by using the IP address and static port (NodePort) on each node. A NodePort service routes to a ClusterIP service, which is automatically created. You can access the NodePort service outside the cluster by requesting`<NodeIP>:<NodePort>`.
  - Server Load Balancer: The Server Load Balancer service, which is provided by Alibaba Cloud. You can configure Internet access or intranet access by using this type of service. Server Load Balancer can route to the NodePort service and ClusterIP service.
- **Port Mapping**: Add a service port and a container port. If you select NodePort for **Type**, you must configure a node port to avoid port conflicts. TCP and UDP protocols are supported.

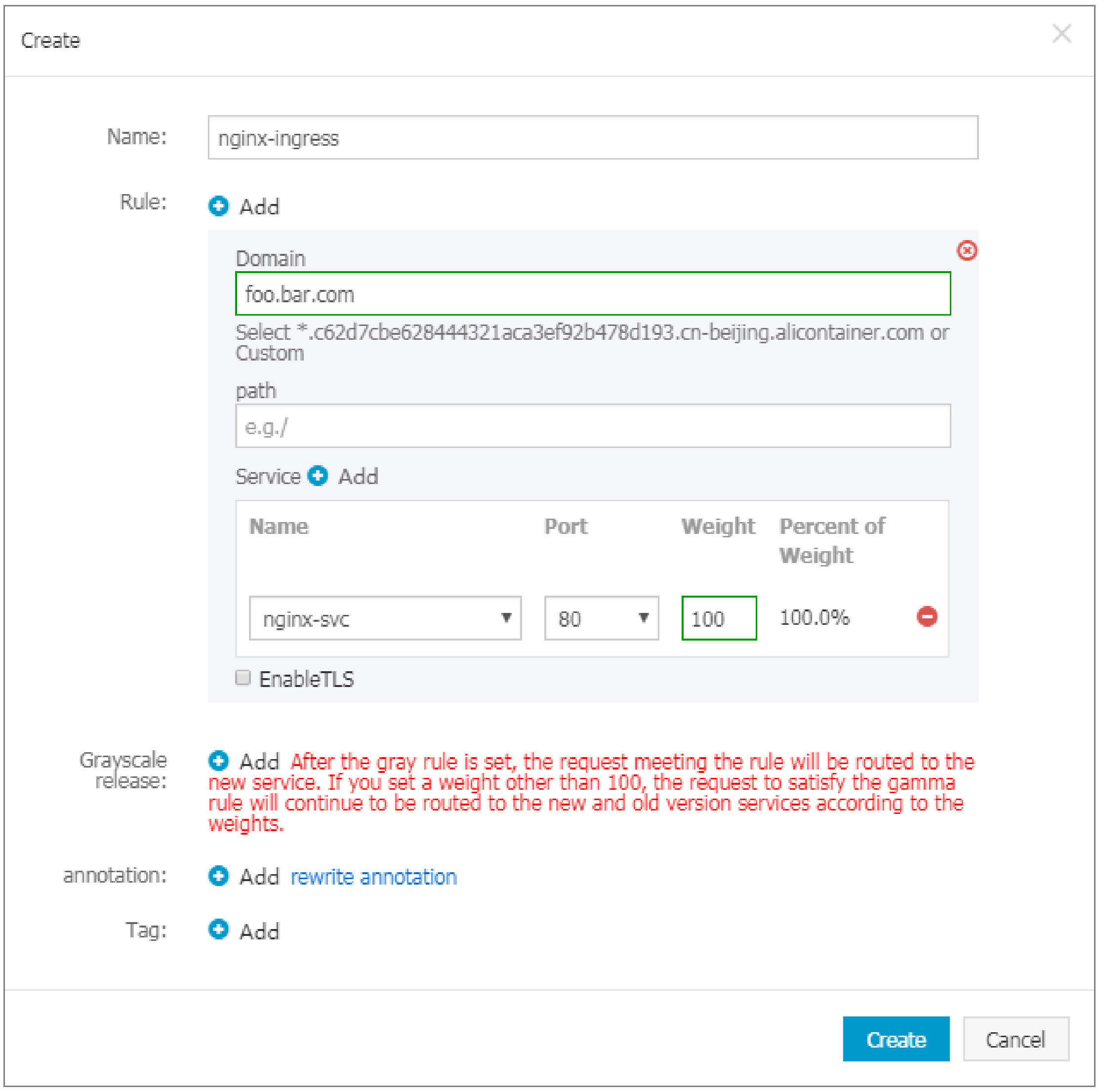
- **annotation**: Add an annotation to the service. Server Load Balancer configuration parameters are supported, see *Access services by using Server Load Balancer*.
- **Label**: You can add a label to the service to identify the service.

2. Click **Create** at the right of Ingress. Configure rout rules for the backend pod in the displayed dialog box, and then click **Create**. For more information about route configuration, see *Ingress configurations*.

Note:

When you create an application by using an image, you can create ingress for only one service. In this example, use a virtual host name as the testing domain name. You need to add a record to the hosts. In actual work scenarios, use a filing domain name.

```
101.37.224.146   foo.bar.com    #the IP address of ingress
```



Create

Name: nginx-ingress

Rule: Add

Domain: foo.bar.com

Select *.c62d7cbe628444321aca3ef92b478d193.cn-beijing.alicontainer.com or Custom

path: e.g./

Service Add

Name	Port	Weight	Percent of Weight
nginx-svc	80	100	100.0%

EnableTLS

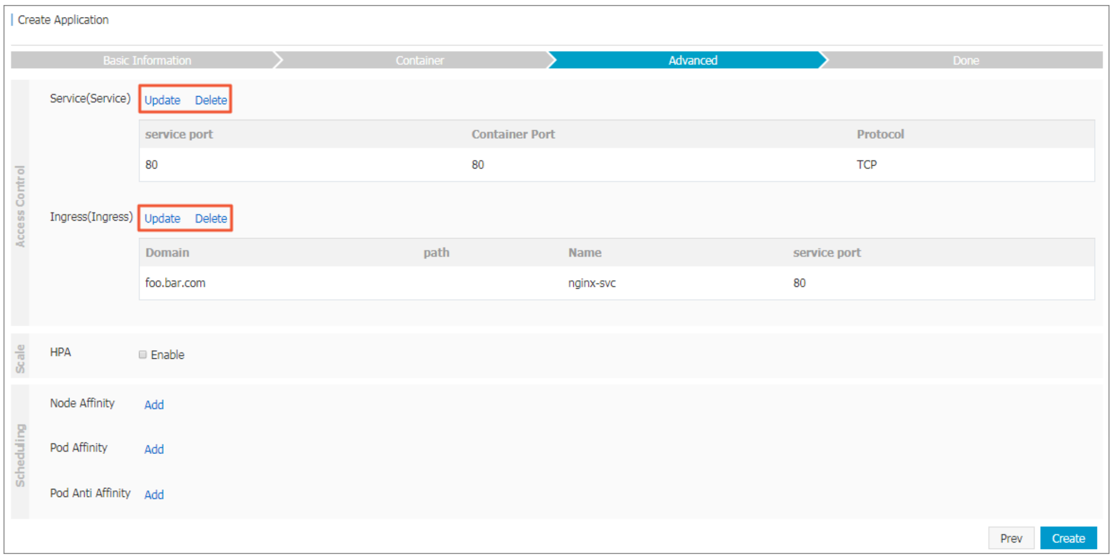
Grayscale release: Add After the gray rule is set, the request meeting the rule will be routed to the new service. If you set a weight other than 100, the request to satisfy the gamma rule will continue to be routed to the new and old version services according to the weights.

annotation: Add rewrite annotation

Tag: Add

Create Cancel

3. The created service and ingress are displayed in the access control section. You can reconfigure the service and ingress by clicking **Update** and **Delete**.



Service(Service) Update Delete

service port	Container Port	Protocol
80	80	TCP

Ingress(Ingress) Update Delete

Domain	path	Name	service port
foo.bar.com		nginx-svc	80

HPA: Enable

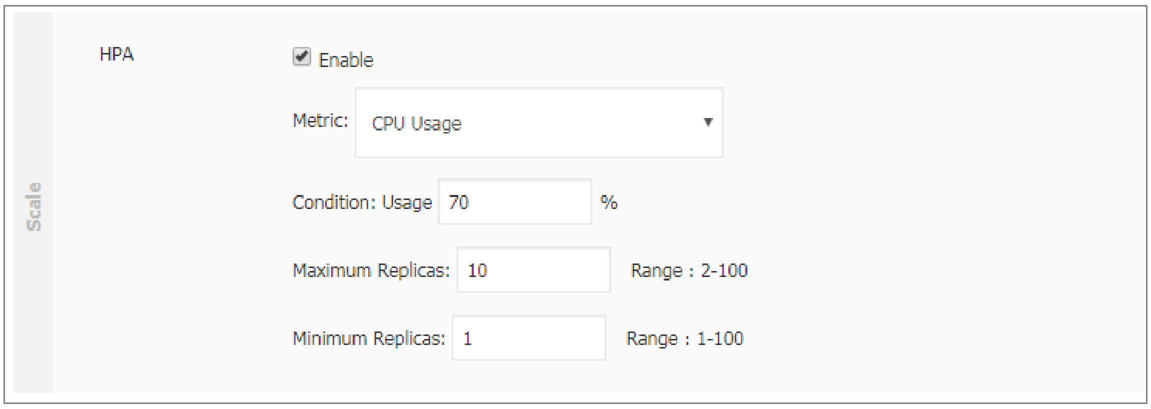
Node Affinity: Add

Pod Affinity: Add

Pod Anti Affinity: Add

b) Optional: Configure **Horizontal Pod Autoscaling (HPA)**.

You can choose whether to enable **HPA**. To meet the demands of applications under different loads, Container Service supports the container auto scaling, which automatically adjusts the number of containers according to the container CPU and memory usage.



HPA ✓ Enable

Metric: CPU Usage

Condition: Usage 70 %

Maximum Replicas: 10 Range : 2-100

Minimum Replicas: 1 Range : 1-100

Note:
To enable auto scaling, you must configure required resources for the deployment. Otherwise, the container auto scaling cannot take effect. See the basic configuration of containers.

- **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.

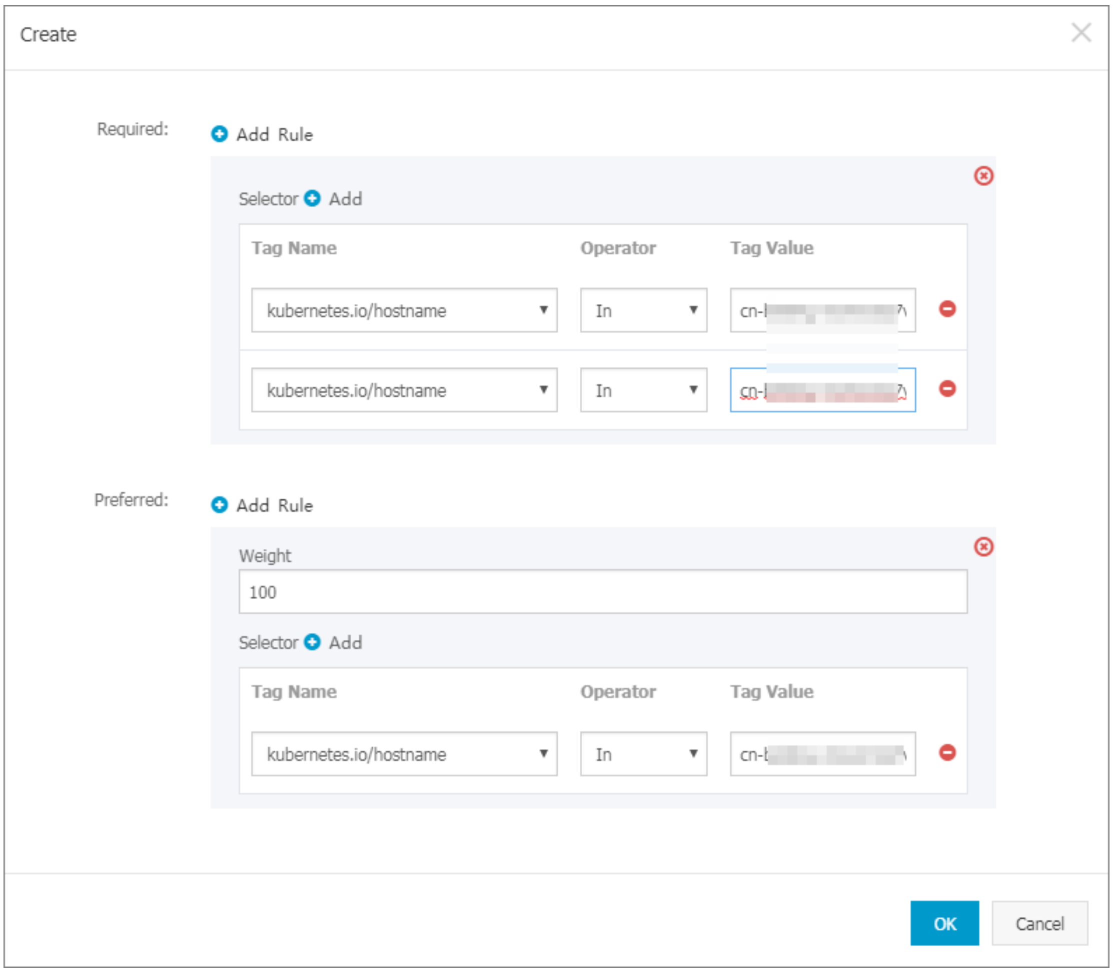
- **Maximum Replicas**: The maximum number of replicas that the deployment can expand to.
- **Minimum Replicas**: The minimum number of replicas that the deployment can contract to.

c) Optional: Configure **Scheduling Affinity**.

You can configure node affinity, pod affinity, and pod anti affinity. For more information, see *https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity*.

Note:
Affinity scheduling depends on node tags and pod tags. You can use built-in tads to schedule as well as configure tags for nodes and pods in advance.

1. Set **Node Affinity** by configuring node tags.


Create

Required: Add Rule

Selector Add

Tag Name	Operator	Tag Value
kubernetes.io/hostname	In	cn-[redacted]
kubernetes.io/hostname	In	cn-[redacted]

Preferred: Add Rule

Weight: 100

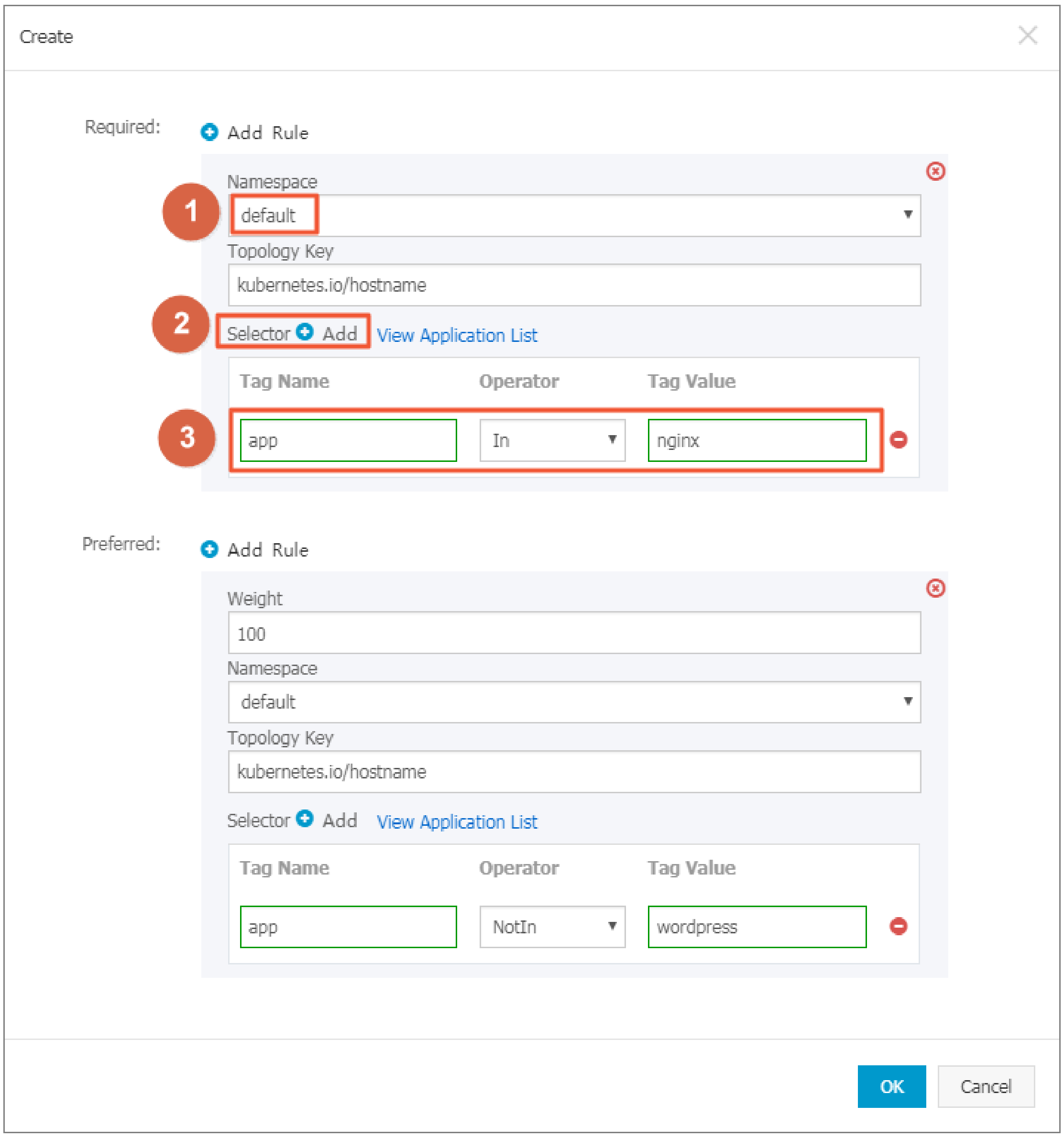
Selector Add

Tag Name	Operator	Tag Value
kubernetes.io/hostname	In	cn-[redacted]

Node scheduling supports both required and preferred rules, and various operators such as 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 nodes with corresponding tags. You can add multiple required rules, but you only need to meet one of them.
- **Preferred** rules are not necessary satisfied and correspond to **`preferredDuringSchedulingIgnoredDuringExecution`**. In this example, the schedule tries not to schedule the pod to the node with the corresponding tag. You can also set weights for preferred rules. If multiple nodes that match the criteria exist, the node with the highest weight is scheduled as a priority. You can define multiple preferred rules, but all rules must be satisfied before scheduling.

2. Configure **Pod Affinity** to deploy the pod of the application in a topology domain together with other pods. For example, services that communicate with each other can be deployed to the same topology domain (such as a host) by configuring pod affinity scheduling to reduce network latency between them.


Create

Required: Add Rule

Namespace: default

Topology Key: kubernetes.io/hostname

Selector Add View Application List

Tag Name	Operator	Tag Value
app	In	nginx

Preferred: Add Rule

Weight: 100

Namespace: default

Topology Key: kubernetes.io/hostname

Selector Add View Application List

Tag Name	Operator	Tag Value
app	NotIn	wordpress

Schedule pods according to tags of pods running on nodes. Available expressions are `In, NotIn, Exists, DoesNotExist`.

- **Required** rules must be satisfied and correspond to **`requiredDuringSchedulingIgnoredDuringExecution`**. The pod affinity scheduling must meet configured rules.
  - **Namespace**: The scheduling policy is based on pod tags so it is constrained by namespaces.
  - **Topology Key**: Specifies the domain to be scheduled through tags of nodes. For example, if you set `kubernetes.io/hostname` as the topology key, nodes are used to identify topologies. If you specify`beta.kubernetes.io/os` as the topology key, operating systems of nodes are used to identify topologies.

- **Selector**: By clicking the Add button at the right of Selector, you can add hard constraint rules.
- **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 this affinity configuration dialog box.
- Hard constraints: Configure tags of existing applications, operators, and tag values. In this example, schedule the application to be created to this host that runs applications with the `app: nginx` tag.

• **Preferred** rules, that is, soft constraints, corresponding to **`preferredDuringSchedulingIgnoredDuringExecution`**. The pod affinity scheduling meet configured rules as soon as possible. For soft constraint rules, you can configure the weight of each rule. Other configuration requirements are the same as hard constraint rules.

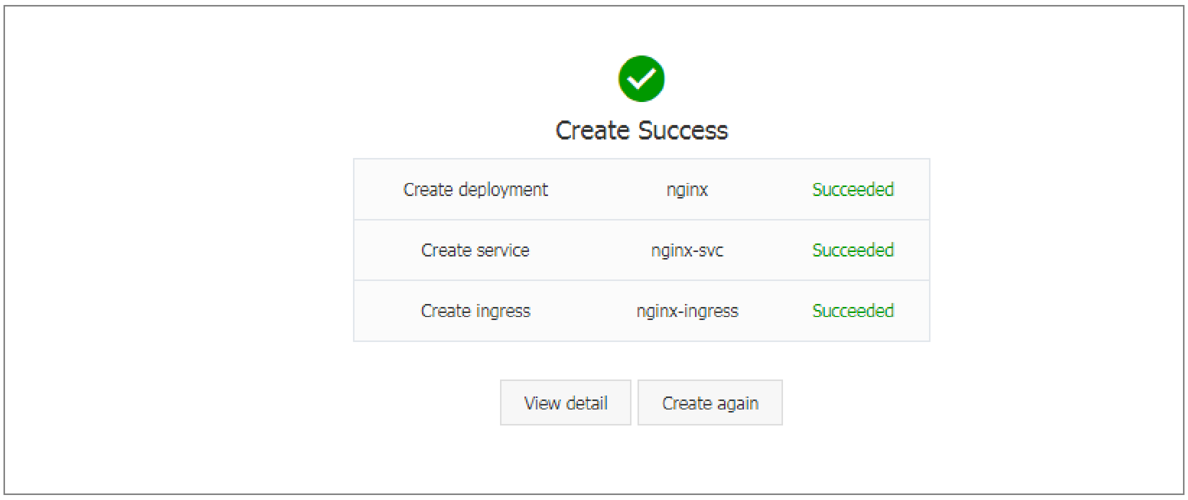
Note:
Weight: Specifies the weight of one soft constraint rule in the range of 1 to 100. Weights of nodes that satisfies configured soft constraint rules are calculated through algorithm and then the pod is scheduled to the node with the greatest weight.

3. Configure **Pod Anti Affinity** to deploy the pod of the application in a topology domain excluding other pods. Scenarios that use pod anti affinity scheduling include:

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

Note:
Configuration methods of pod anti affinity scheduling are the same as that of pod affinity. But the same scheduling rules have different meanings for pod anti affinity scheduling. Select an appropriate scheduling rule based on scenarios.

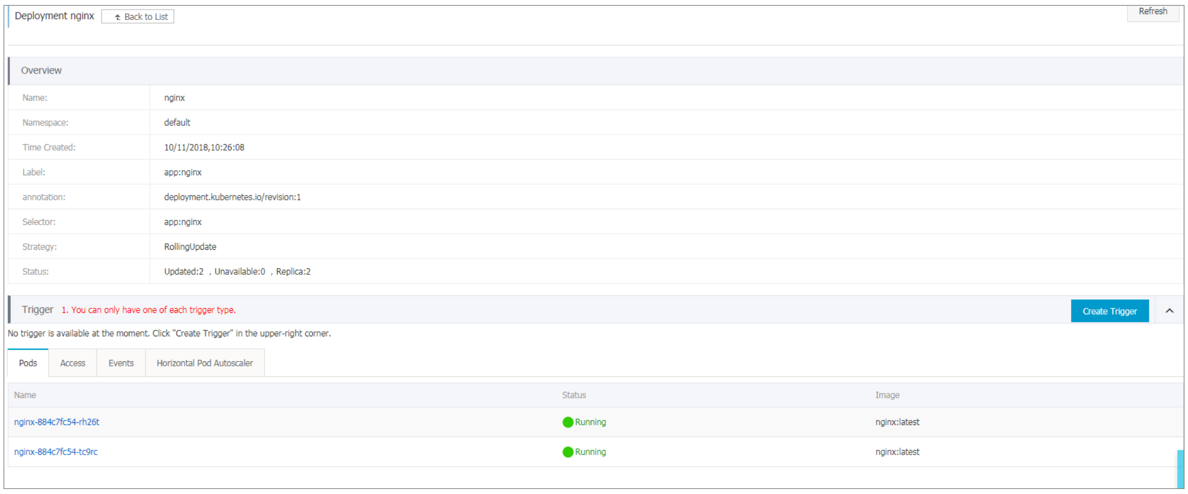
7. Click **Create**.
8. After you create the application, the create success page is displayed and objects contained in the application are listed by default. You can click **View detail** to view the deployment details.



Create Success

Create deployment	nginx	Succeeded
Create service	nginx-svc	Succeeded
Create ingress	nginx-ingress	Succeeded

The nginx-deployment page is displayed by default.



Overview

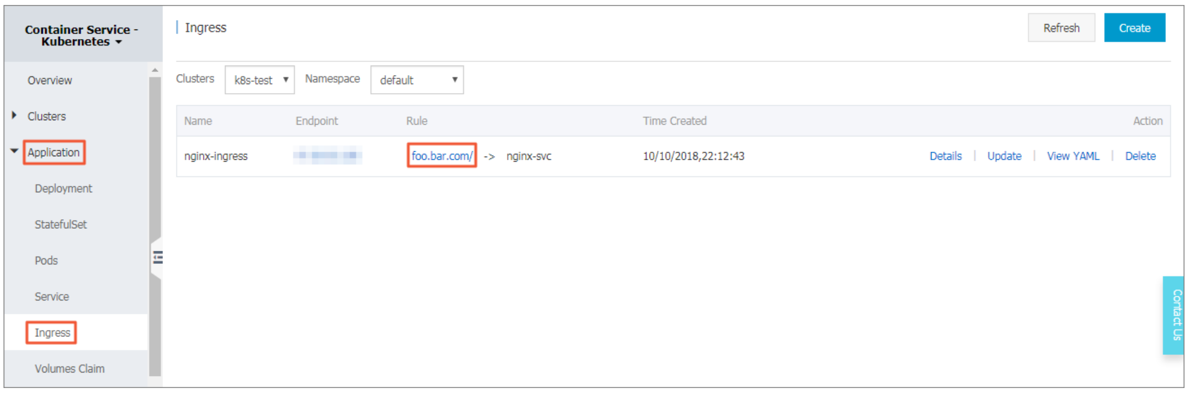
Name:	nginx
Namespace:	default
Time Created:	10/11/2018,10:26:08
Label:	app:nginx
annotation:	deployment.kubernetes.io/revision:1
Selector:	app:nginx
Strategy:	RollingUpdate
Status:	Updated:2 , Unavailable:0 , Replica:2

Trigger 1. You can only have one of each trigger type.

No trigger is available at the moment. Click "Create Trigger" in the upper-right corner.

Name	Status	Image
nginx-884c7fc54-rh26t	● Running	nginx:latest
nginx-884c7fc54-tc9rc	● Running	nginx:latest

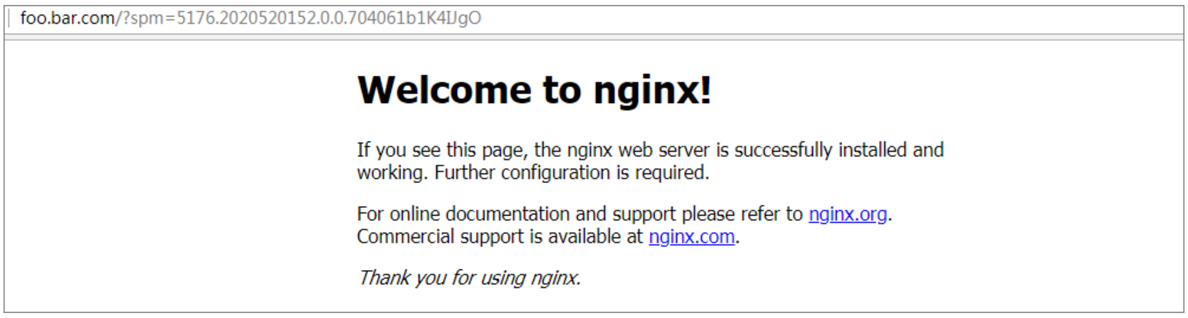
9. Click **Application** > **Ingress** in the left-side navigation pane, a rule is displayed under the Ingress list.



Clusters k8s-test Namespace default

Name	Endpoint	Rule	Time Created	Action
nginx-ingress	[redacted]	foo.bar.com/ -> nginx-svc	10/10/2018,22:12:43	Details | Update | View YAML | Delete

10. Access the Ingress testing domain in a browser and you can see that the Nginx welcome page is displayed.


foo.bar.com/?spm=5176.2020520152.0.0.704061b1K4IJgO

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

1.6.2 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 storage 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:

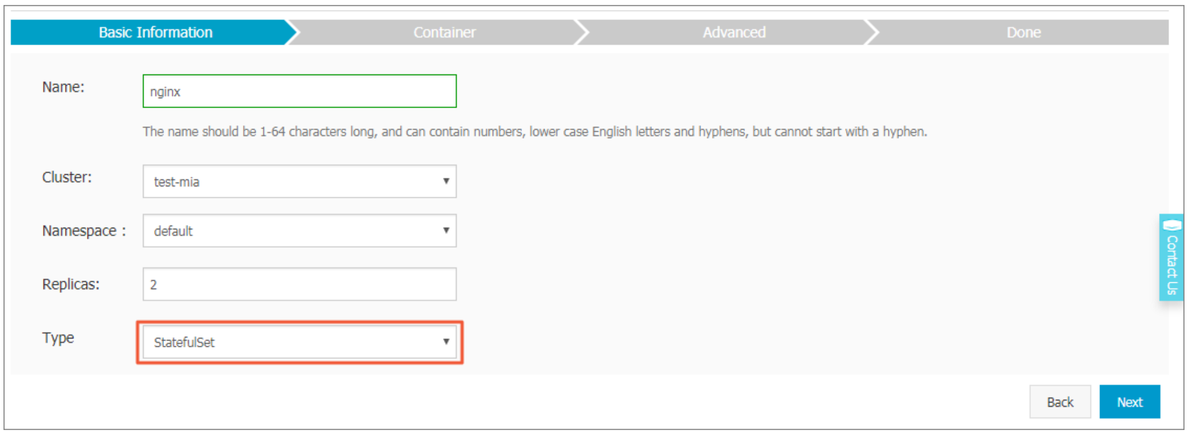
Scenarios	Description
Pod consistency	Contains order (such as startup and stop order) and network consistency. This consistency is related to pods and has nothing to do with the node to which the pods are to be scheduled.
Stable persistent storage	Create a PV for each pod through VolumeClai mTemplate. Deleting or reducing replicas does not delete relevant volumes.
Stable network marker	The hostname mode for a pod is: `(statefulset 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. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane, and then click **Create by 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.



Name: nginx

The name should be 1-64 characters long, and can contain numbers, lower case English letters and hyphens, but cannot start with a hyphen.

Cluster: test-mia

Namespace : default

Replicas: 2

Type StatefulSet

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/*.

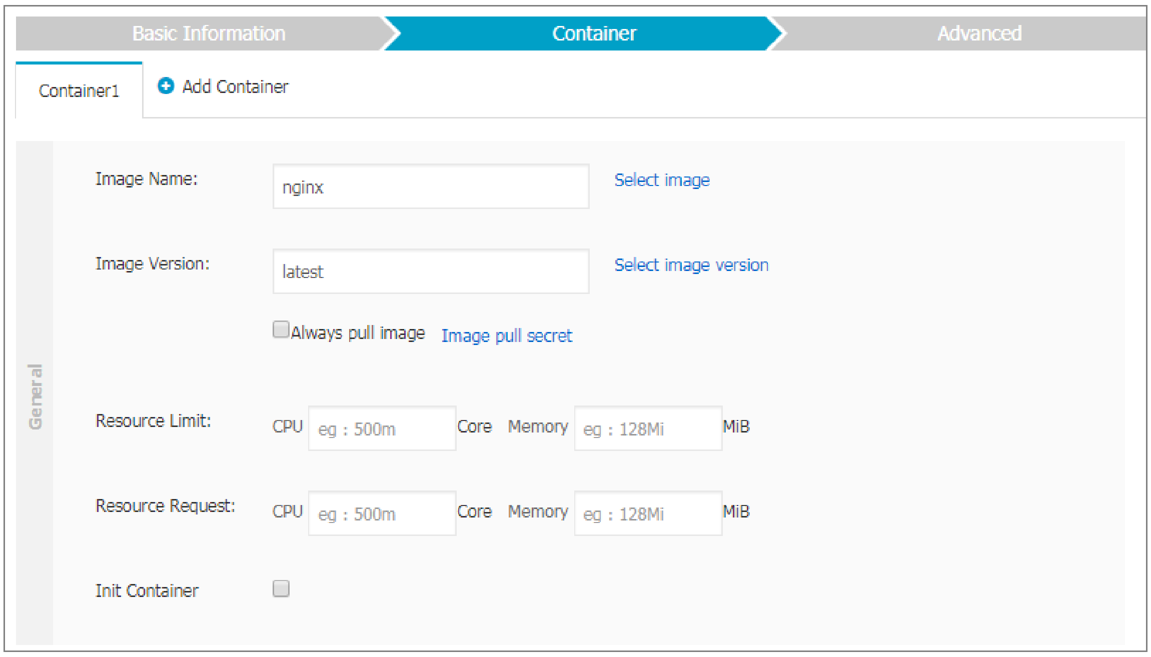


Image Name: nginx Select image

Image Version: latest Select image version

Always pull image Image pull secret

Resource Limit: CPU eg : 500m Core Memory eg : 128Mi MiB

Resource Request: CPU eg : 500m Core Memory eg : 128Mi MiB

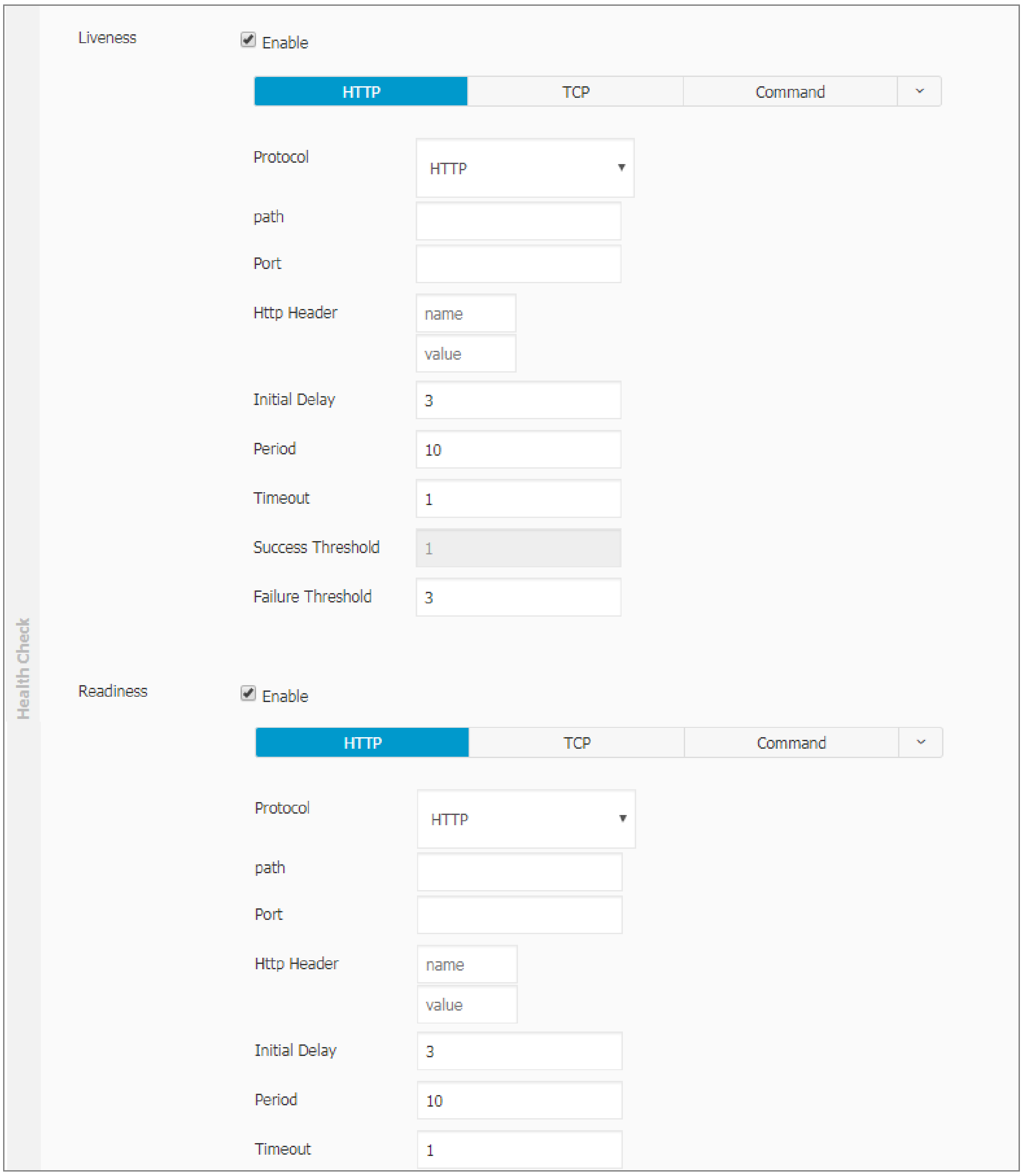
Init Container

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*.



Liveness ✓ Enable

Protocol: HTTP
path:
Port:
Http Header: name / value
Initial Delay: 3
Period: 10
Timeout: 1
Success Threshold: 1
Failure Threshold: 3

Readiness ✓ Enable

Protocol: HTTP
path:
Port:
Http Header: name / value
Initial Delay: 3
Period: 10
Timeout: 1

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.

Request method	Description
	• HTTP Header: Custom headers in the HTTP request. HTTP allows repeated headers. Supports the correct configurat ion 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.
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

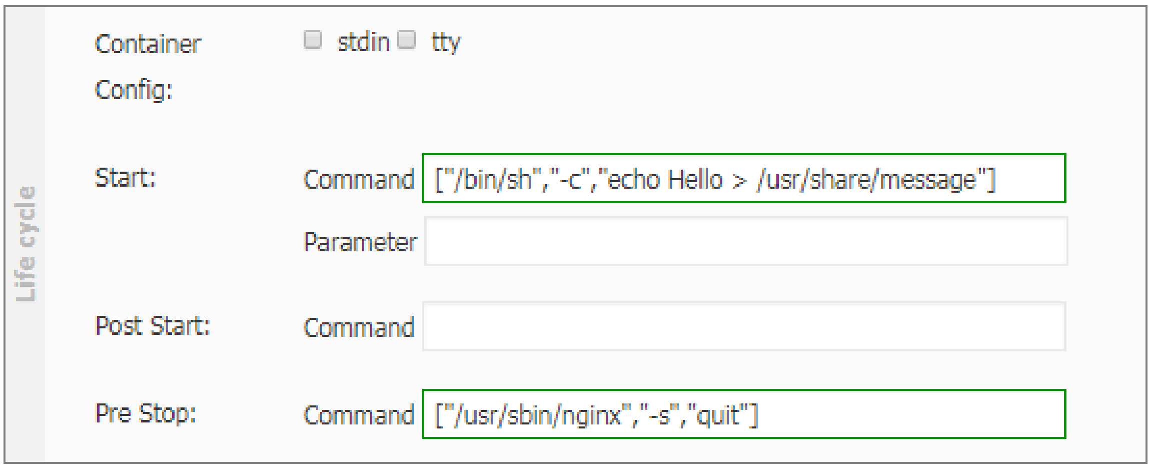
Request method	Description
	wait after the container is started. The default is 15. • Period (in seconds): Namely, the **periodseconds**. Intervals at which the probe is performed. The default value is 10. The minimum value is 1. • Timeout (in seconds): Namely, the **timeoutSeconds**. The time of probe 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.
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.

Request method	Description
	• 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: 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**: Select the stdin check box to enable standard input for the container. Select the tty check box to assign an virtual terminal to for the container to send signals to the container. These two options are usually used together, which indicates to bind the terminal (tty) to the container standard input (stdin). For example, an interactive program obtains standard input from you and then displays the obtained standard input in the terminal.
- **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.



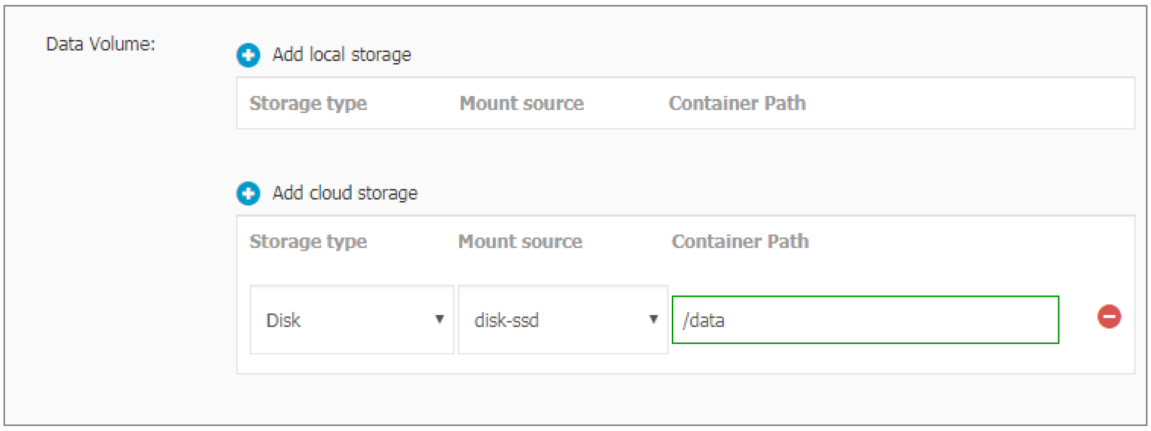
Container Config:	☐ stdin ☐ tty	
Start:	Command	["/bin/sh","-c","echo Hello > /usr/share/message"]
	Parameter	
Post Start:	Command	
Pre Stop:	Command	["/usr/sbin/nginx","-s","quit"]

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.



Storage type	Mount source	Container Path

Storage type	Mount source	Container Path
Disk	disk-ssd	/data

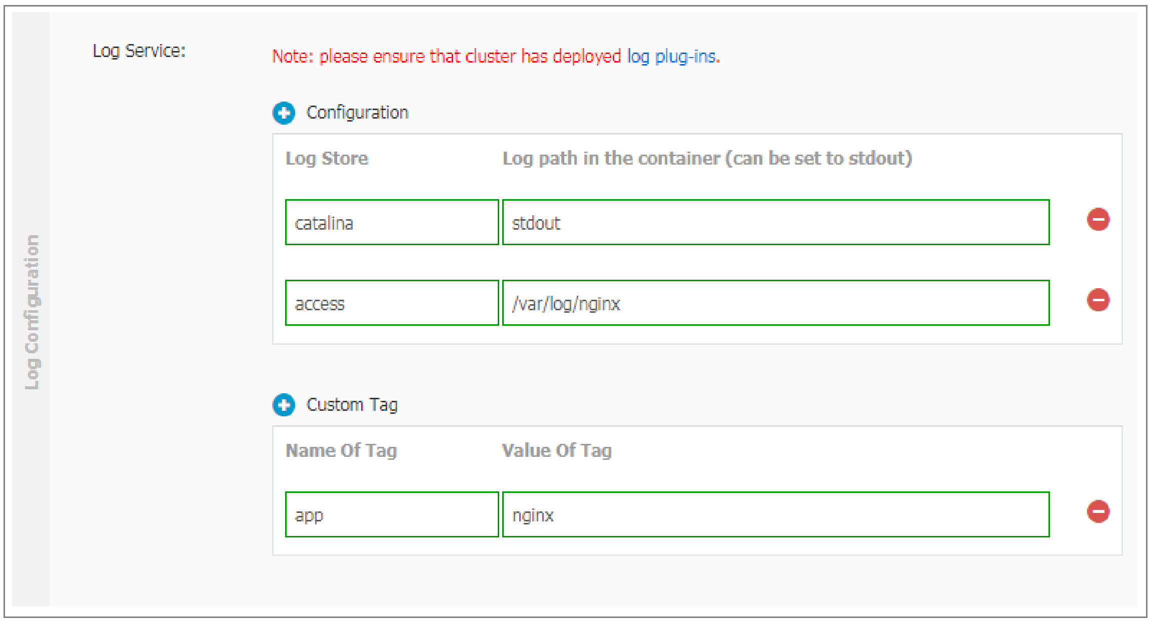
f) Optional: Configure **Log Service**. You can configure collection methods and customize tags for this service.

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 convenience to log analysis such as log statistics and filter.



Note: please ensure that cluster has deployed log plug-ins.

Log Store	Log path in the container (can be set to stdout)
catalina	stdout
access	/var/log/nginx

Name Of Tag	Value Of Tag
app	nginx

5. Click **Next** after completing the configurations.
6. Configure advanced settings. In this example, configure only access settings.

   a) Configure **Access Control**.

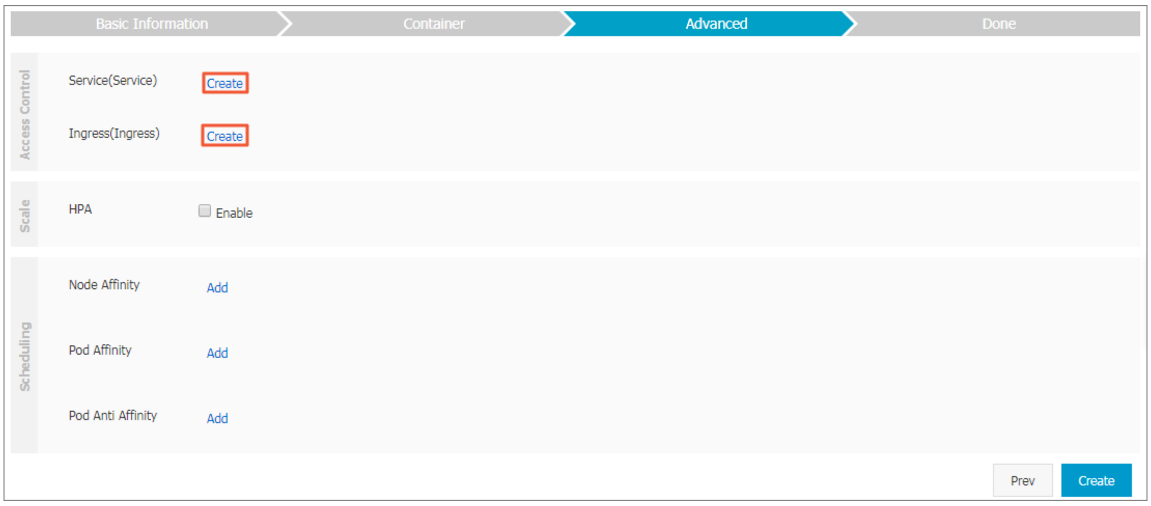
   You can configure how to expose the backend pod and click **Create**. In this example, select Cluster IP Service and Ingress to create an nginx application that is accessible for Internet.

Note:

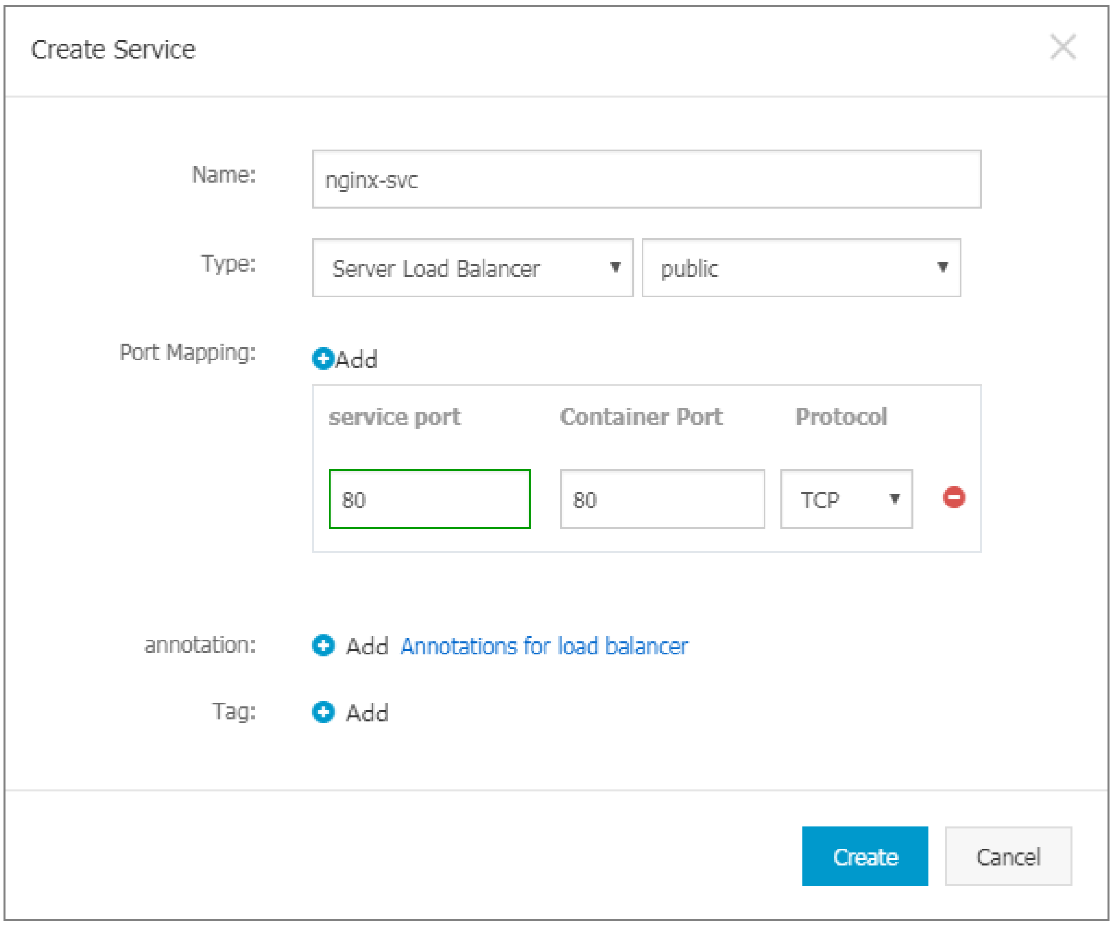
To meet communication requirements of the application, you can configure access control based on your needs:

- Internal applications: For applications that work only inside a cluster, you can create services of Cluster IP or Node Port for internal communication as needed.
- External applications: For applications that need to be exposed to Internet, you can configure access control by using one of the following methods:
  - Create a service of Server Load Balancer: Use the Server Load Balancer (SLB) service provided by Alibaba Cloud, which provides Internet accessibility for the application.

- Create a service of ClusterIP or NodePort, and create Ingress: This method provides Internet accessibility through ingress. For more information, see *https://kubernetes.io/docs/concepts/services-networking/ingress/*.



1. Click **Create** at right of Service. Configure a service in the displayed dialog box, and then click **Create**.


Create Service

Name: nginx-svc

Type: Server Load Balancer, public

Port Mapping:

service port	Container Port	Protocol
80	80	TCP

- **Name**: You can enter your custom name. The default is `applicationname-svc`.

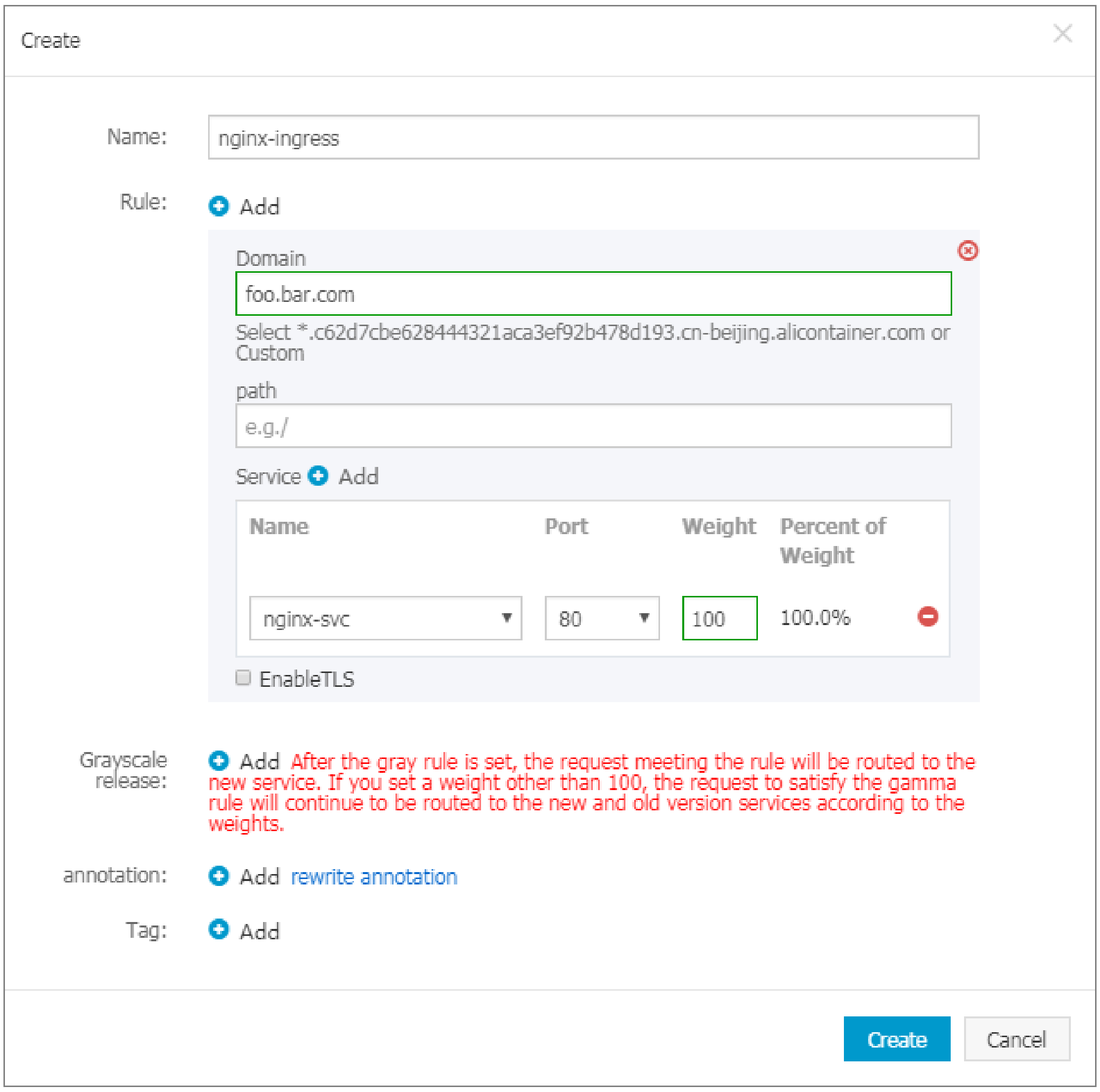
- **Type**: Select one from the following three service types.
  - ClusterIP: Expose the service by using the internal IP address of your cluster. With this type selected, the service is accessible only within the cluster.
  - NodePort: Expose the service by using the IP address and static port (NodePort) on each node. A NodePort service routes to a ClusterIP service, which is automatically created. You can access the NodePort service outside the cluster by requesting`<NodeIP>:<NodePort>`.
  - Server Load Balancer: The Server Load Balancer service, which is provided by Alibaba Cloud. You can configure Internet access or intranet access by using this type of service. Server Load Balancer can route to the NodePort service and ClusterIP service.
- **Port Mapping**: Add a service port and a container port. If you select NodePort for **Type**, you must configure a node port to avoid port conflicts. TCP and UDP protocols are supported.
- **annotation**: Add an annotation to the service. Server Load Balancer configuration parameters are supported, see *Access services by using Server Load Balancer*.
- **Label**: You can add a label to the service to identify the service.

2. Click **Create** at the right of Ingress. Configure rout rules for the backend pod in the displayed dialog box, and then click **Create**. For more information about route configuration, see *Ingress configurations*.

Note:

When you create an application by using an image, you can create ingress for only one service. In this example, use a virtual host name as the testing domain name. You need to add a record to the hosts. In actual work scenarios, use a filing domain name.

```
101.37.224.146   foo.bar.com    #the IP address of ingress
```


Create

Name: nginx-ingress

Rule: Add

Domain

foo.bar.com

Select *.c62d7cbe628444321aca3ef92b478d193.cn-beijing.alicontainer.com or Custom

path

e.g./

Service Add

Name	Port	Weight	Percent of Weight
nginx-svc	80	100	100.0%

EnableTLS

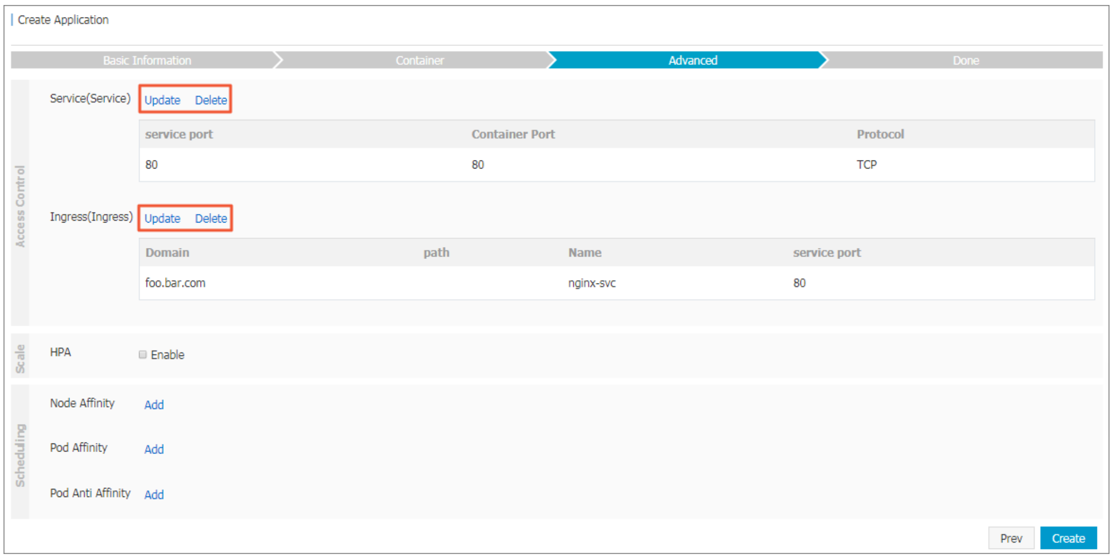
Grayscale release: Add After the gray rule is set, the request meeting the rule will be routed to the new service. If you set a weight other than 100, the request to satisfy the gamma rule will continue to be routed to the new and old version services according to the weights.

annotation: Add rewrite annotation

Tag: Add

Create Cancel

3. The created service and ingress are displayed in the access control section. You can reconfigure the service and ingress by clicking **Update** and **Delete**.



Service(Service)

service port	Container Port	Protocol
80	80	TCP

Ingress(Ingress)

Domain	path	Name	service port
foo.bar.com		nginx-svc	80

HPA ☐ Enable

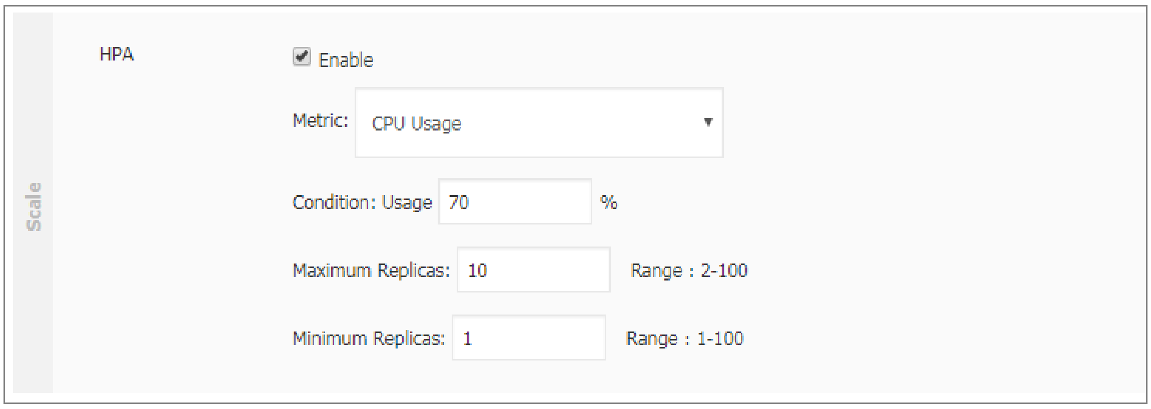
Node Affinity

Pod Affinity

Pod Anti Affinity

b) Optional: Configure **Horizontal Pod Autoscaling (HPA)**.

You can choose whether to enable **HPA**. To meet the demands of applications under different loads, Container Service supports the container auto scaling, which automatically adjusts the number of containers according to the container CPU and memory usage.



HPA ✓ Enable

Metric: CPU Usage

Condition: Usage 70 %

Maximum Replicas: 10 Range : 2-100

Minimum Replicas: 1 Range : 1-100

Note:

To enable auto scaling, you must configure required resources for the deployment. Otherwise, the container auto scaling cannot take effect. See the basic configuration of containers.

- **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.

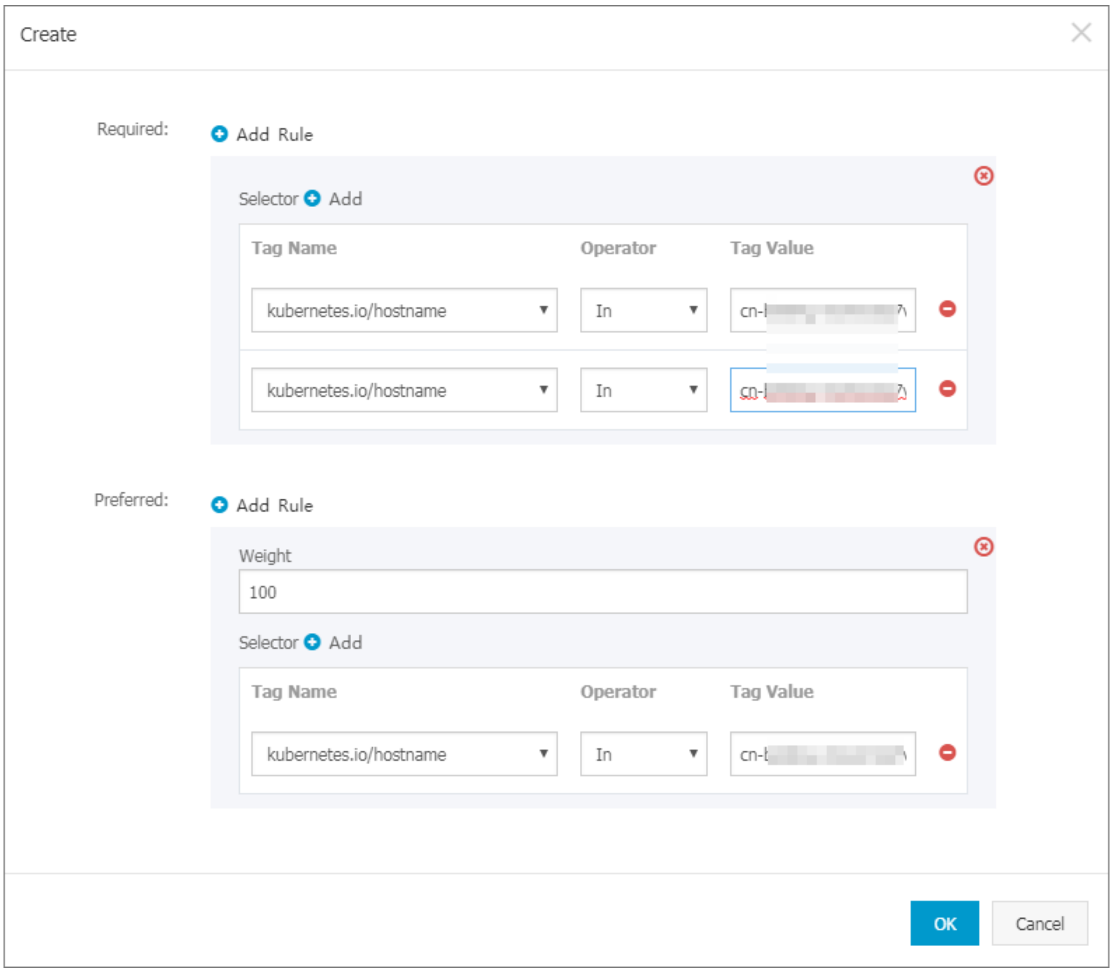
- **Minimum Replicas**: The minimum number of replicas that the deployment can contract to.

c) Optional: Configure **Scheduling Affinity**.

You can configure node affinity, pod affinity, and pod anti affinity. For more information, see *https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity*.

Note:
Affinity scheduling depends on node tags and pod tags. You can use built-in tads to schedule as well as configure tags for nodes and pods in advance.

1. Set **Node Affinity** by configuring node tags.



Create

Required: Add Rule

Selector Add

Tag Name	Operator	Tag Value
kubernetes.io/hostname	In	cn-[redacted]
kubernetes.io/hostname	In	cn-[redacted]

Preferred: Add Rule

Weight
100

Selector Add

Tag Name	Operator	Tag Value
kubernetes.io/hostname	In	cn-[redacted]

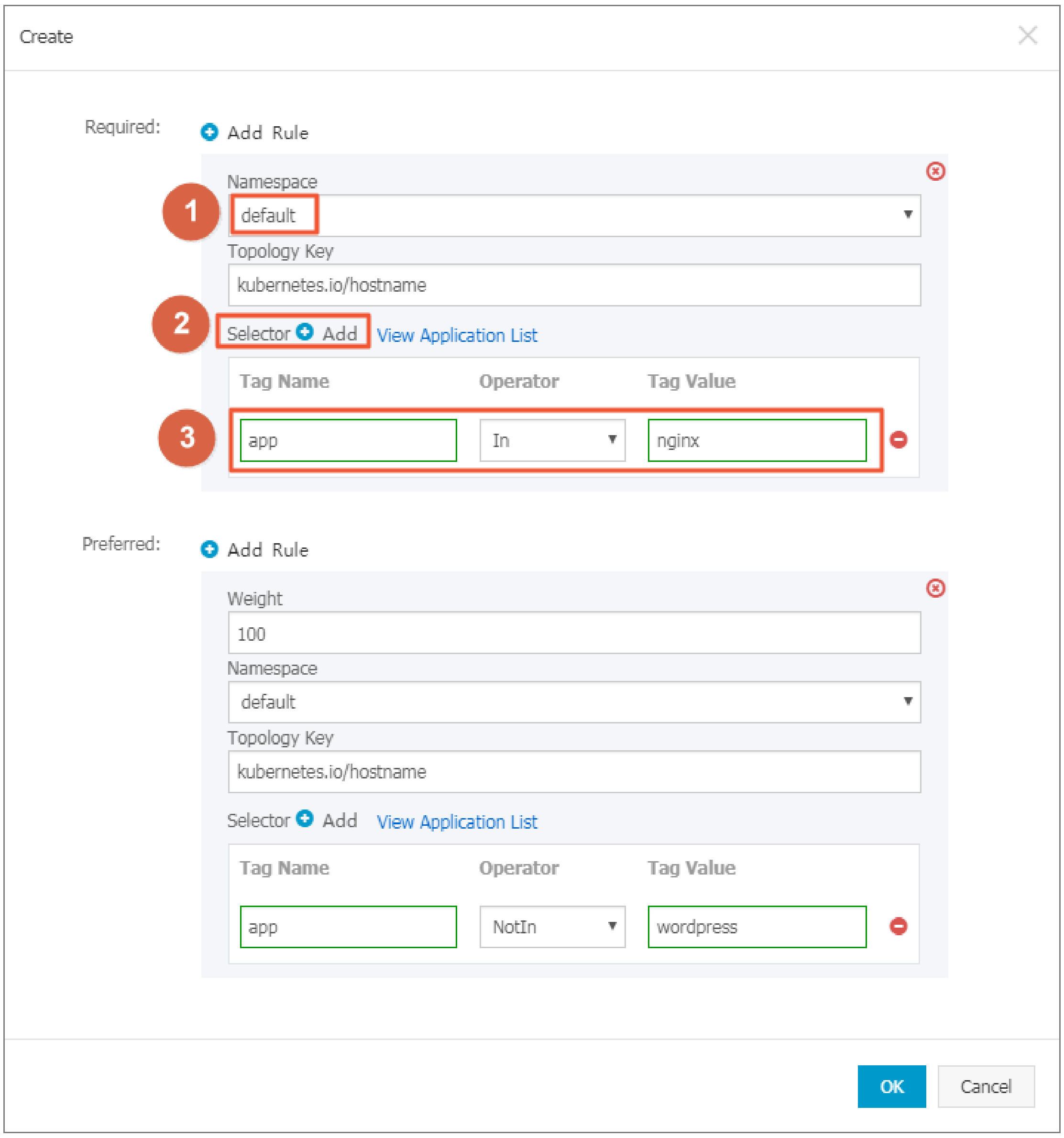
OK Cancel

Node scheduling supports both required and preferred rules, and various operators such as 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 nodes with corresponding tags. You can add multiple required rules, but you only need to meet one of them.

- **Preferred** rules are not necessary satisfied and correspond to **`preferredDuringSchedulingIgnoredDuringExecution`**. In this example, the schedule tries not to schedule the pod to the node with the corresponding tag. You can also set weights for preferred rules. If multiple nodes that match the criteria exist, the node with the highest weight is scheduled as a priority. You can define multiple preferred rules, but all rules must be satisfied before scheduling.

2. Configure **Pod Affinity** to deploy the pod of the application in a topology domain together with other pods. For example, services that communicate with each other can be deployed to the same topology domain (such as a host) by configuring pod affinity scheduling to reduce network latency between them.


Create

Required: Add Rule

Namespace: default

Topology Key: kubernetes.io/hostname

Selector Add View Application List

Tag Name	Operator	Tag Value
app	In	nginx

Preferred: Add Rule

Weight: 100

Namespace: default

Topology Key: kubernetes.io/hostname

Selector Add View Application List

Tag Name	Operator	Tag Value
app	NotIn	wordpress

Schedule pods according to tags of pods running on nodes. Available expressions are `In, NotIn, Exists, DoesNotExist`.

- **Required** rules must be satisfied and correspond to **`requiredDuringSchedulingIgnoredDuringExecution`**. The pod affinity scheduling must meet configured rules.
  - **Namespace**: The scheduling policy is based on pod tags so it is constrained by namespaces.
  - **Topology Key**: Specifies the domain to be scheduled through tags of nodes. For example, if you set `kubernetes.io/hostname` as the topology key, nodes are used to identify topologies. If you specify`beta.kubernetes.io/os` as the topology key, operating systems of nodes are used to identify topologies.
  - **Selector**: By clicking the Add button at the right of Selector, you can add hard constraint rules.
  - **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 this affinity configuration dialog box.
  - Hard constraints: Configure tags of existing applications, operators, and tag values. In this example, schedule the application to be created to this host that runs applications with the `app: nginx` tag.
- **Preferred** rules, that is, soft constraints, corresponding to **`preferredDuringSchedulingIgnoredDuringExecution`**. The pod affinity scheduling meet configured rules as soon as possible. For soft constraint rules, you can configure the weight of each rule. Other configuration requirements are the same as hard constraint rules.

  Note:

  Weight: Specifies the weight of one soft constraint rule in the range of 1 to 100. Weights of nodes that satisfies configured soft constraint rules are calculated through algorithm and then the pod is scheduled to the node with the greatest weight.

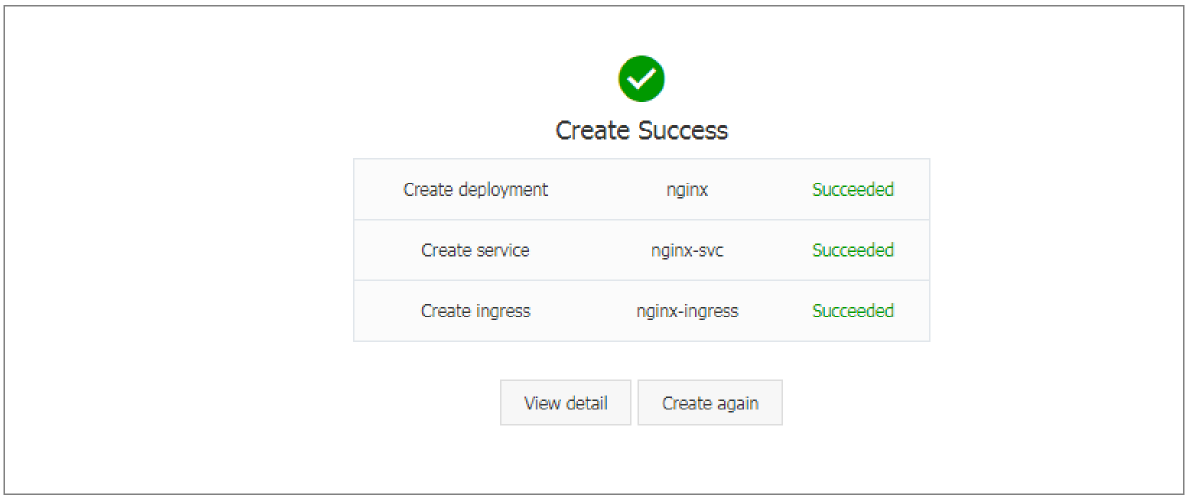
3. Configure **Pod Anti Affinity** to deploy the pod of the application in a topology domain excluding other pods. Scenarios that use pod anti affinity scheduling include:

- Distribute pods of a service to different topology domains (such as hosts) to improve the stability of the service.

- Grant a pot the exclusive access to a node so as to guarantee no other pods use resources of the node.
- Distribute pods of services that may affect each other to different hosts.

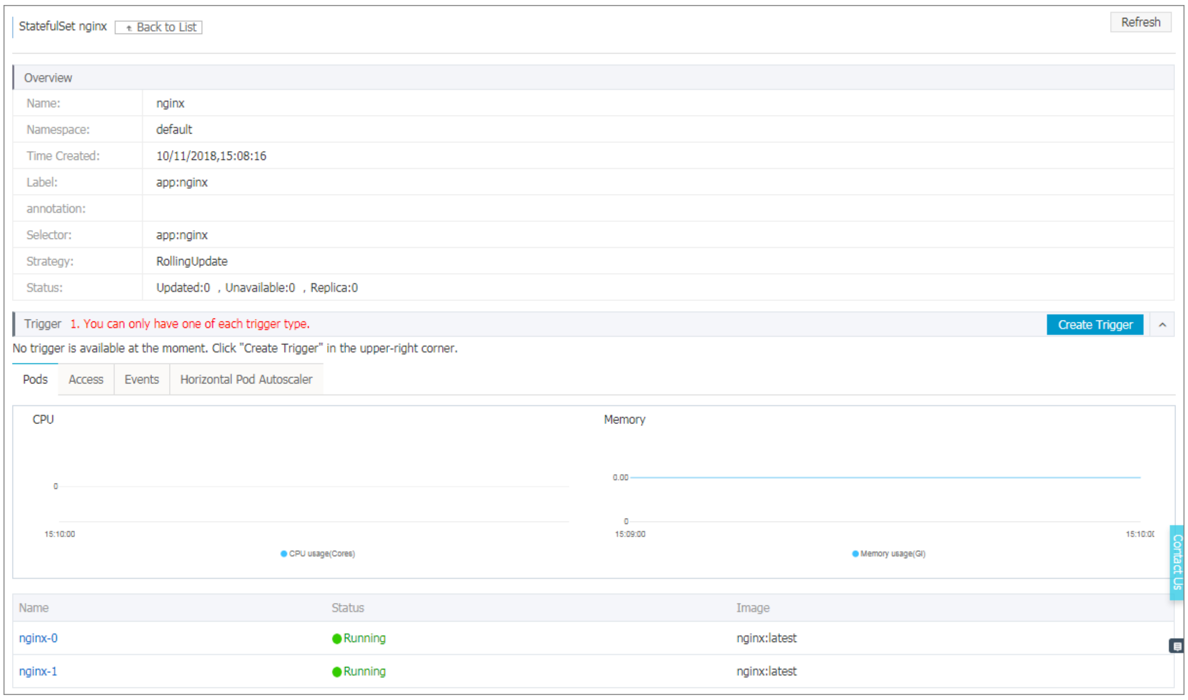
Note:
Configuration methods of pod anti affinity scheduling are the same as that of pod affinity. But the same scheduling rules have different meanings for pod anti affinity scheduling. Select an appropriate scheduling rule based on scenarios.

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.


Create Success

Create deployment	nginx	Succeeded
Create service	nginx-svc	Succeeded
Create ingress	nginx-ingress	Succeeded

The StatefulSet page is displayed by default.



Overview

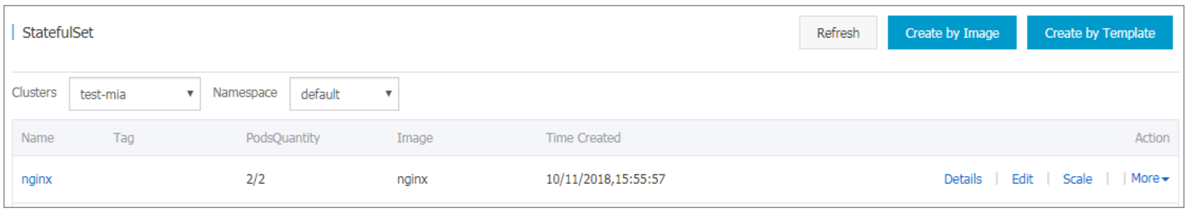
Name:	nginx
Namespace:	default
Time Created:	10/11/2018,15:08:16
Label:	app:nginx
annotation:	
Selector:	app:nginx
Strategy:	RollingUpdate
Status:	Updated:0 , Unavailable:0 , Replica:0

Trigger 1. You can only have one of each trigger type.

No trigger is available at the moment. Click "Create Trigger" in the upper-right corner.

Name	Status	Image
nginx-0	● Running	nginx:latest
nginx-1	● Running	nginx:latest

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

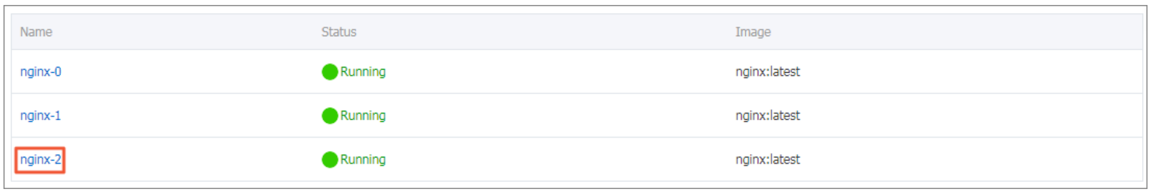


Clusters test-mia Namespace default

Name	Tag	PodsQuantity	Image	Time Created	Action
nginx		2/2	nginx	10/11/2018,15:55:57	Details | Edit | Scale | More

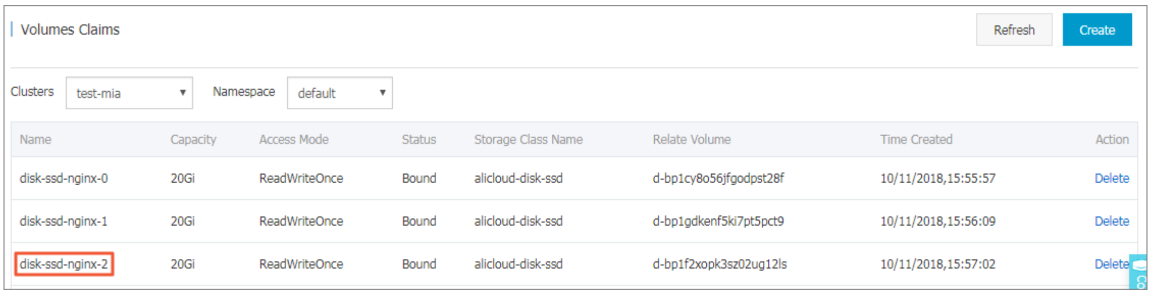
10. Optional: To verify service scalability, click **Scale** at the right of a target nginx application.

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.



Name	Status	Image
nginx-0	● Running	nginx:latest
nginx-1	● Running	nginx:latest
nginx-2	● Running	nginx:latest

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.



Clusters: test-mia Namespace: default

Name	Capacity	Access Mode	Status	Storage Class Name	Relate Volume	Time Created	Action
disk-ssd-nginx-0	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd	d-bp1cy8o56jfgodpst28f	10/11/2018,15:55:57	Delete
disk-ssd-nginx-1	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd	d-bp1gdkenf5ki7pt5pct9	10/11/2018,15:56:09	Delete
disk-ssd-nginx-2	20Gi	ReadWriteOnce	Bound	alicloud-disk-ssd	d-bp1f2xopk3sz02ug12ls	10/11/2018,15:57:02	Delete

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/statefulset         #add a temporty
file named statefulset

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

Remove the pod to verify the data persistence:

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

# kubectl exec nginx-1 ls /tmp                         #data
persistence storage
lost+found
statefulset
```

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.

1.6.3 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.

1.6.4 Create an 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

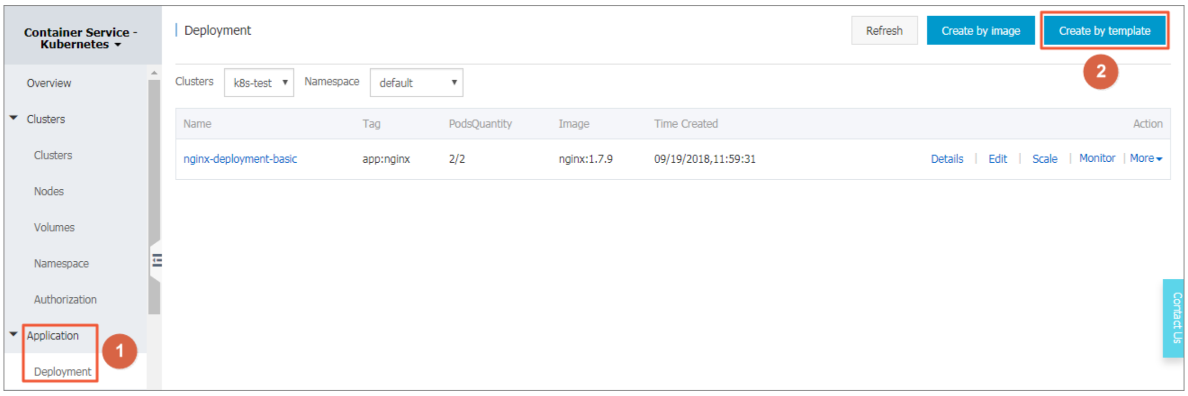
Create a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

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

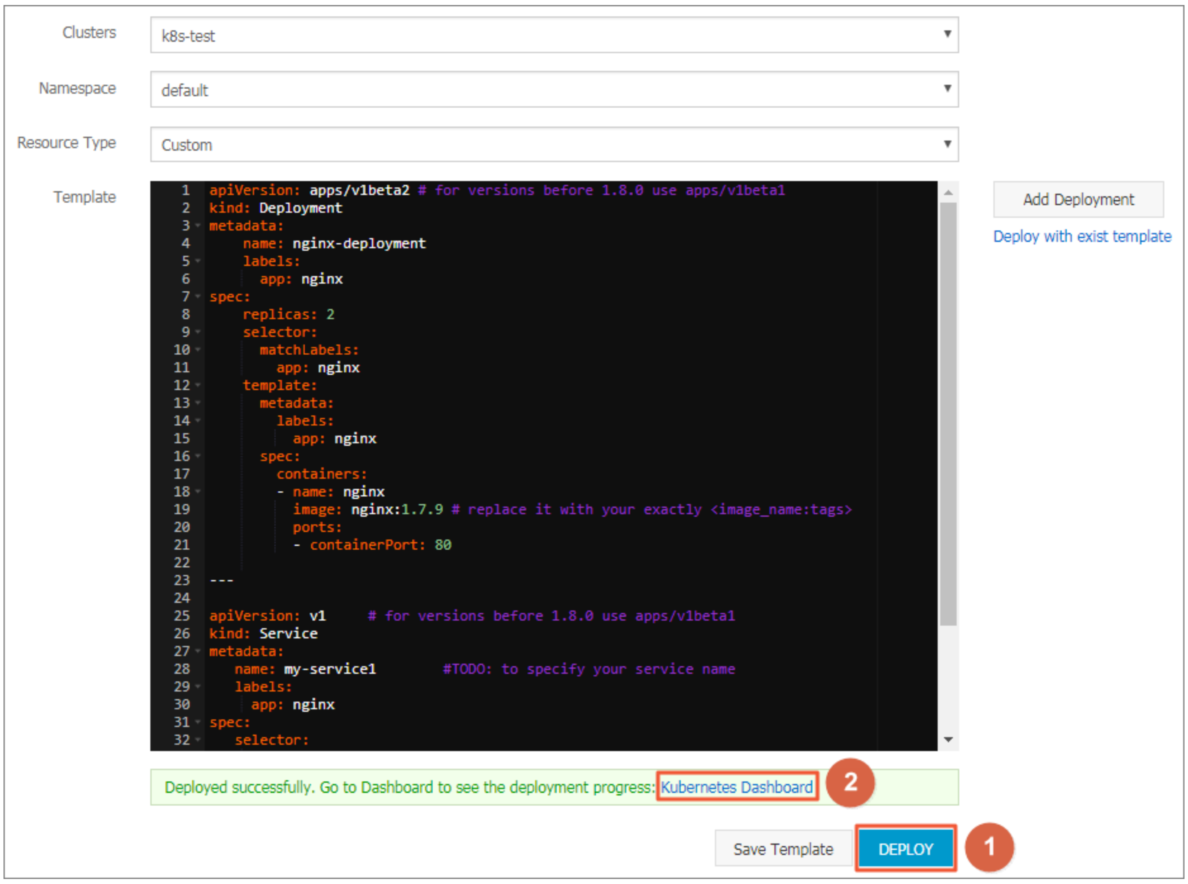
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane.
3. Click **Create by template** in the upper-right corner.



Name	Tag	PodsQuantity	Image	Time Created	Action
nginx-deployment-basic	app:nginx	2/2	nginx:1.7.9	09/19/2018,11:59:31	Details | Edit | Scale | Monitor | More

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.



Clusters: k8s-test

Namespace: default

Resource Type: Custom

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:
      matchLabels:
        app: nginx
    template:
      metadata:
        labels:
          app: nginx
      spec:
        containers:
        - name: nginx
          image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
          ports:
          - containerPort: 80

---

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:
```

Deployed successfully. Go to Dashboard to see the deployment progress: Kubernetes Dashboard

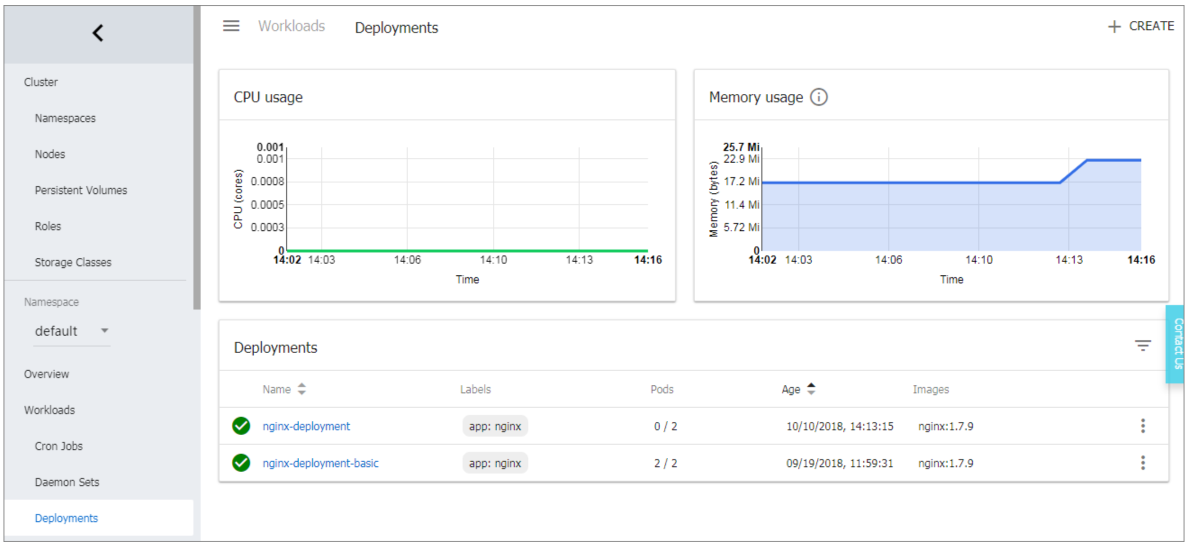
The following is a sample orchestration for an Nginx application. The orchestration is based on an orchestration template built in Container Service. By using this orchestration template, you can create a deployment that belongs to an Nginx application quickly.

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:
      matchLabels:
        app: nginx
    template:
      metadata:
        labels:
          app: nginx
      spec:
        containers:
        - name: nginx
          image: nginx:1.7.9 # replace it with your exactly <
image_name:tags>
          ports:
          - containerPort: 80

---

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: LoadBalancer        ##In this example, change the type from
 NodePort to LoadBalancer.
```

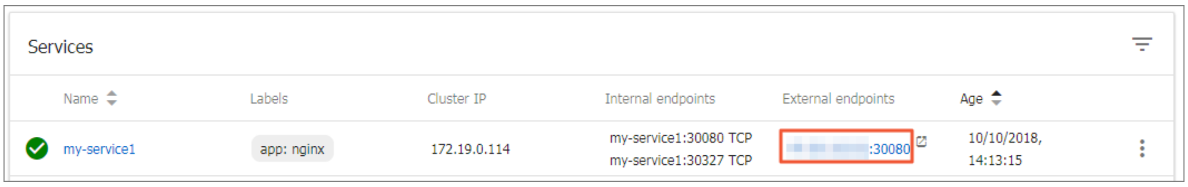
5. After you click **DEPLOY**, a message indicating the deployment status is displayed. After the deployment succeeds, click **Kubernetes Dashboard** in the message to go to the dashboard and check the deployment progress.



Deployments

Name	Labels	Pods	Age	Images
nginx-deployment	app: nginx	0 / 2	10/10/2018, 14:13:15	nginx:1.7.9
nginx-deployment-basic	app: nginx	2 / 2	09/19/2018, 11:59:31	nginx:1.7.9

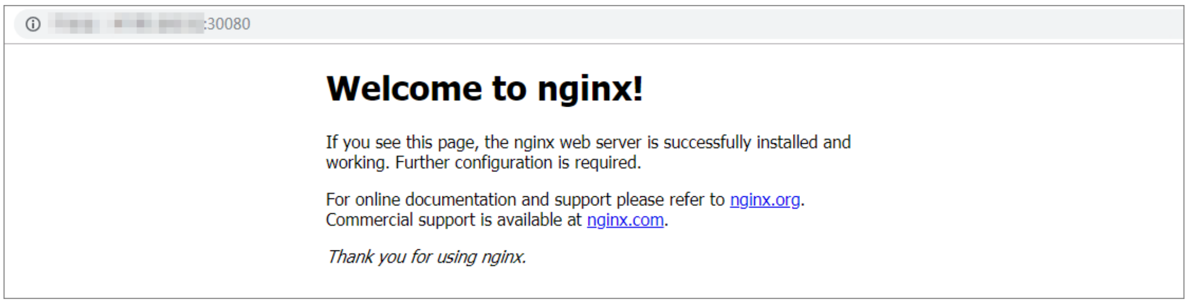
6. In the Kubernetes dashboard, you can see that the service named my-service1 is successfully deployed and its external endpoint is exposed. Click the access address under **External endpoints**.



Services

Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
my-service1	app: nginx	172.19.0.114	my-service1:30080 TCP my-service1:30327 TCP	:30080	10/10/2018, 14:13:15

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



:30080

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

What's next

You can also go back to the home page of Container Services and then click **Application** > **Services** in the left-side navigation pane to view the Nginx service.

1.6.5 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_48*.

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=LoadBalancer` 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=LoadBalancer
```

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
```

1.6.6 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 demonstrates 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 facilitates 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

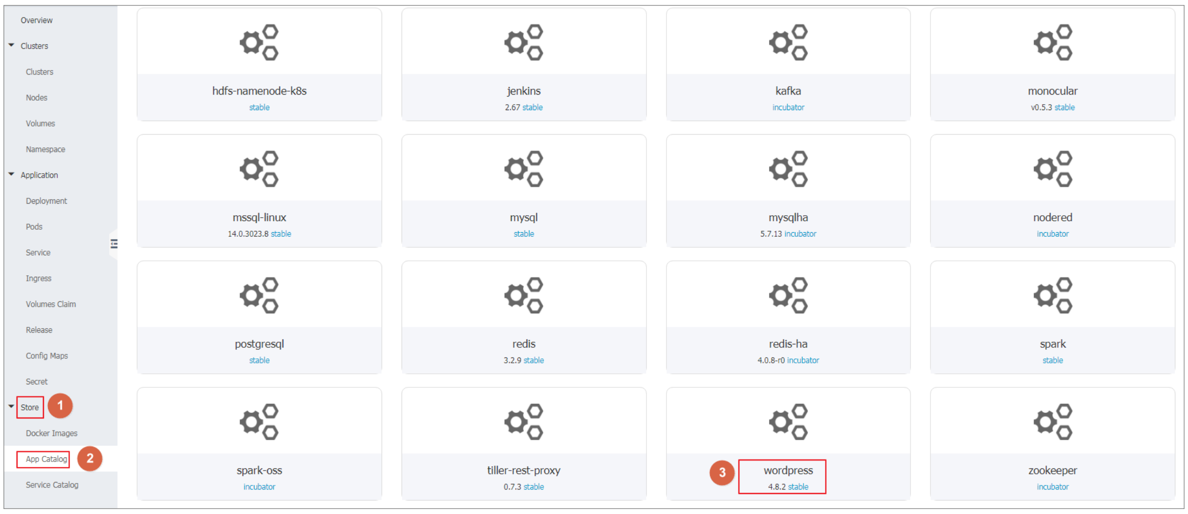
- Before using Helm to deploy an application, create a Kubernetes cluster in Alibaba Cloud Container Service. For more information, see *Create a Kubernetes cluster*.

  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.

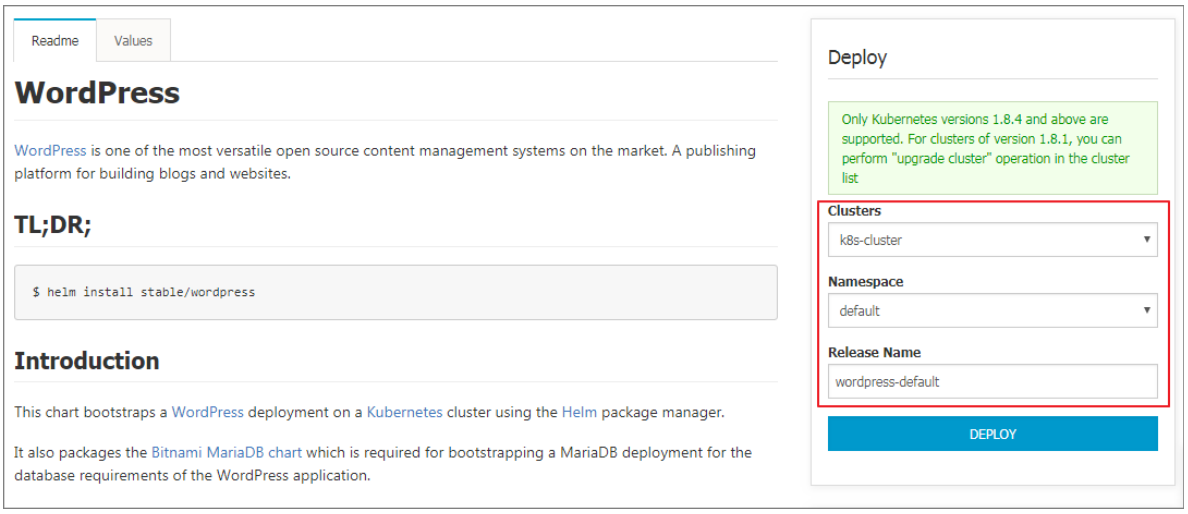
Deploy applications in Container Service console

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.


Readme Values

WordPress

WordPress is one of the most versatile open source content management systems on the market. A publishing platform for building blogs and websites.

TL;DR;

```
$ helm install stable/wordpress
```

Introduction

This chart bootstraps a WordPress deployment on a Kubernetes cluster using the Helm package manager.

It also packages the Bitnami MariaDB chart which is required for bootstrapping a MariaDB deployment for the database requirements of the WordPress application.

Deploy

Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

Clusters: k8s-cluster

Namespace: default

Release Name: wordpress-default

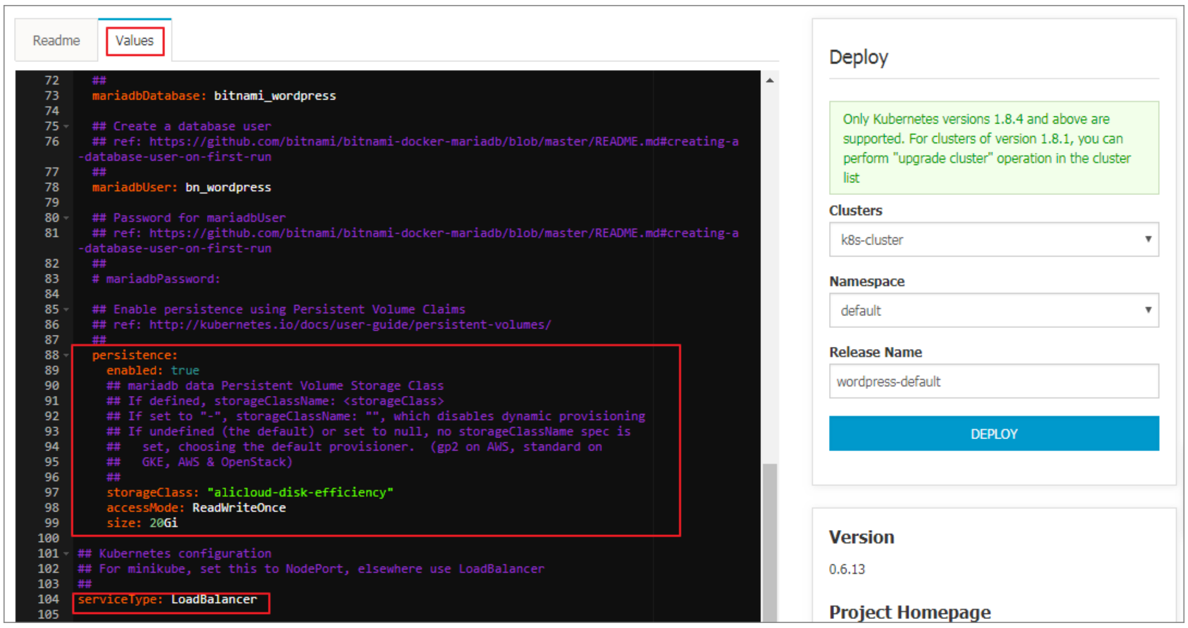
DEPLOY

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.



Readme | Values

```
  ##
  mariadbDatabase: bitnami_wordpress

  ## Create a database user
  ## ref: https://github.com/bitnami/bitnami-docker-mariadb/blob/master/README.md#creating-a
-database-user-on-first-run
  ##
  mariadbUser: bn_wordpress

  ## Password for mariadbUser
  ## ref: https://github.com/bitnami/bitnami-docker-mariadb/blob/master/README.md#creating-a
-database-user-on-first-run
  ##
  # mariadbPassword:

  ## Enable persistence using Persistent Volume Claims
  ## ref: http://kubernetes.io/docs/user-guide/persistent-volumes/
  ##
  persistence:
    enabled: true
    ## mariadb data Persistent Volume Storage Class
    ## If defined, storageClassName: <storageClass>
    ## If set to "-", storageClassName: "", which disables dynamic provisioning
    ## If undefined (the default) or set to null, no storageClassName spec is
    ##   set, choosing the default provisioner.  (gp2 on AWS, standard on
    ##   GKE, AWS & OpenStack)
    ##
    storageClass: "alicloud-disk-efficiency"
    accessMode: ReadWriteOnce
    size: 20Gi

## Kubernetes configuration
## For minikube, set this to NodePort, elsewhere use LoadBalancer
##
serviceType: LoadBalancer

```

Deploy

Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

Clusters: k8s-cluster

Namespace: default

Release Name: wordpress-default

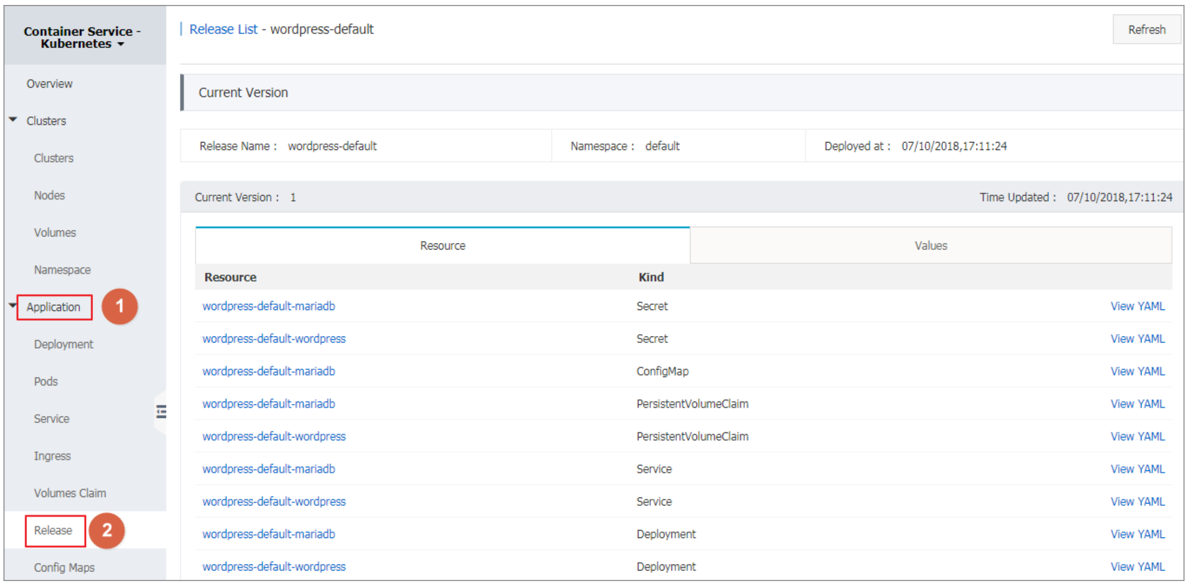
DEPLOY

Version

0.6.13

Project Homepage

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



Release List - wordpress-default

Current Version

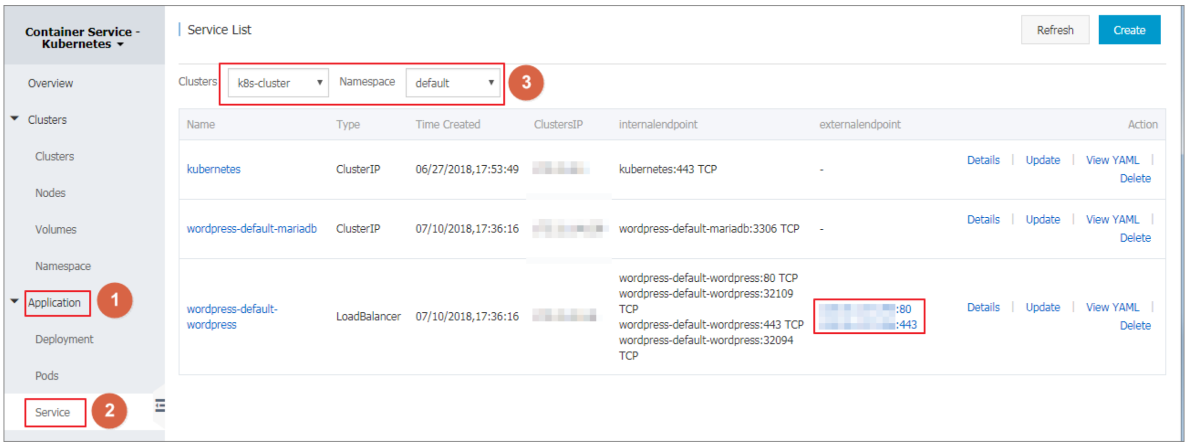
Release Name : wordpress-default | Namespace : default | Deployed at : 07/10/2018,17:11:24

Current Version : 1 | Time Updated : 07/10/2018,17:11:24

Resource | Values

Resource	Kind	
wordpress-default-mariadb	Secret	View YAML
wordpress-default-wordpress	Secret	View YAML
wordpress-default-mariadb	ConfigMap	View YAML
wordpress-default-mariadb	PersistentVolumeClaim	View YAML
wordpress-default-wordpress	PersistentVolumeClaim	View YAML
wordpress-default-mariadb	Service	View YAML
wordpress-default-wordpress	Service	View YAML
wordpress-default-mariadb	Deployment	View YAML
wordpress-default-wordpress	Deployment	View YAML

7. Click **Application** > **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.



Service List

Clusters k8s-cluster Namespace default

Name	Type	Time Created	ClustersIP	internalendpoint	externalendpoint	Action
kubernetes	ClusterIP	06/27/2018,17:53:49	[redacted]	kubernetes:443 TCP	-	Details | Update | View YAML | Delete
wordpress-default-mariadb	ClusterIP	07/10/2018,17:36:16	[redacted]	wordpress-default-mariadb:3306 TCP	-	Details | Update | View YAML | Delete
wordpress-default-wordpress	LoadBalancer	07/10/2018,17:36:16	[redacted]	wordpress-default-wordpress:80 TCP wordpress-default-wordpress:32109 TCP wordpress-default-wordpress:443 TCP wordpress-default-wordpress:32094 TCP	[redacted]:80 [redacted]:443	Details | Update | View YAML | Delete

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.loadBalancer.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.
loadBalancer.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/`.

1.6.7 Create a service

A Kubernetes service, which is generally called a microservice, is an abstraction which defines a logical set of pods and a policy by which to access them. The set of pods accessed by a Kubernetes service is usually determined by a Label Selector.

Kubernetes pods are created and deleted in a short time even if they have their own IP addresses . Therefore, using pods directly to provide services externally is not a solution of high availabili ty. The service abstraction decouples the relationship between the frontend and the backend. Therefore, the loose-coupling microservice allows the frontend to not care about the implementa tions of the backend.

For more information, see *Kubernetes service*.

Prerequisite

You have created a Kubernetes cluster successfully. For how to create a Kubernetes cluster, see *#unique_31*.

Step 1 Create a deployment

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application >** > **Deployment in the left-side navigation pane.**Click **Create by template** in the upper-right corner.
3. Select the cluster and namespace to create the deployment. In the Resource Type drop-down list, select Custom to customize the template or a sample template. Then, click **DEPLOY**.

   In this example, the sample template is 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:
     matchLabels:
       app: nginx
   template:
     metadata:
       labels:
         app: nginx
     spec:
       containers:
       - name: nginx
         image: nginx:1.7.9 # replace it with your exactly <
image_name:tags>
         ports:
         - containerPort: 80 ##You must expose this port in the
service.
```

4. Go to the Kubernetes dashboard to view the running status of this deployment.

Step 2 Create a service

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application >** > **Service** 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. Complete the configurations in the displayed Create Service dialog box.

- **Name:** Enter the service name. In this example, enter nginx-svc.
- **Type:** Select the service type, namely, the access method of the service.
  - ClusterIP: Exposes the service by using the internal IP address of your cluster. With this type selected, the service is only accessible from within the cluster. This type is the default service type.
  - NodePort: Exposes the service by using the IP address and static port (NodePort) on each node. A ClusterIP service, to which the NodePort service is routed, is automatically created. You can access the NodePort service from outside the cluster by requesting `<NodeIP>:<NodePort>` .
  - Server Load Balancer: Exposes the service by using Server Load Balancer, which is provided by Alibaba Cloud. Select public or inner to access the service by using the Internet or intranet. Alibaba Cloud Server Load Balancer can route to the NodePort and ClusterIP services.

- **Related deployment:** Select the backend object to bind with this service. In this example, select nginx-deployment-basic, the deployment created in the preceding step. The corresponding Endpoints object is not created if no deployment is selected here. You can manually map the service to your own endpoints. For more information, see *Services without selectors*.
- **Port Mapping:** Add the service port and container port. The container port must be the same as the one exposed in the backend pod.

5. Click **Create**. The nginx-svc service is displayed on the Service List page.

6. View the basic information of the service. Access the external endpoint of the nginx-svc service in the browser.

Then, you have created a service that is related to a backend deployment and accessed the Nginx welcome page successfully.

1.6.8 Service scaling

After an application is created, you can scale out or in the services as per your needs.

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 **Deployments** in the left-side navigation pane to view the created deployments.
5. Click the icon at the right of the deployment and then select **Scale**.

6. The Scale a Deployment dialog box appears. Modify the value of **Desired number of pods** to 2 and then click **OK**.

   Then, a pod is added by expansion and the number of replicas rises to 2.

What's next

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

After the application completes the deployment, you can click a deployment name to view the details of the running Web service. You can view the replica sets in the deployment, and the CPU usage and memory usage of these replica sets. You can also click Pods in the left-side navigation pane, open a pod, and click **LOGS** in the upper-right corner to view the container logs.

Note:
Wait a few minutes if you cannot view any resources.

1.6.9 View services

Context

If the external service is configured when you create the application, in addition to running containers, Kubernetes dashboard creates the external services for pre-assigning the Server Load Balancer to bring traffic to the containers in the cluster.

Procedure

1. Log on to the *Container Service console*.
2. Click Kubernetes >**Application >** > **Service**in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists to view the deployed services.

   You can view the name, type, created time, cluster IP address, internal endpoint, and external endpoint of a service. In this example, you can view the external endpoint (IP address) assigned to the service. Click the IP address to access the Nginx welcome page.

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

1.6.10 Update a service

You can update a service in the Container Service console or Kubernetes dashboard.

Update a service in Container Service console

1. Log on to the *Container Service console*.

2. Under Kubernetes, click **Application >** > **Service** 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 service (nginx-svc in this example).
4. The Update dialog box appears. Modify the template. In this example, change the nodePort to **31000**. Then, click **OK**.
5. On the Service List page, view the changes of the service. In this example, the nodePort is changed as follows.

Update a service in Kubernetes dashboard

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 go to the Kubernetes dashboard.
4. In the Kubernetes dashboard, select the corresponding namespace and click **Services** in the left-side navigation pane.
5. Click the icon at the right of the service and then select **View/edit YAML** from the drop-down list.
6. The Edit a Service dialog box appears. Modify the configurations. In this example, change the nodePort to **31000**. Then, click **UPDATE**.

1.6.11 Delete a service

You can delete a Kubernetes service in the Container Service console.

Prerequisites

- You have created a Kubernetes cluster successfully. For more information, see *#unique_31*.
- You have created a service successfully. For more information, see *#unique_81*.

Procedure

1. Log on to the *Container Service console*.
2. Click Kubernetes >**Application >** > **Service**in the left-side navigation pane.

3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Delete** at the right of the service (nginx-svc in this example).

4. Click **Confirm** in the displayed dialog box. Then, the service is removed from the Service List page.

1.6.12 Use an application trigger

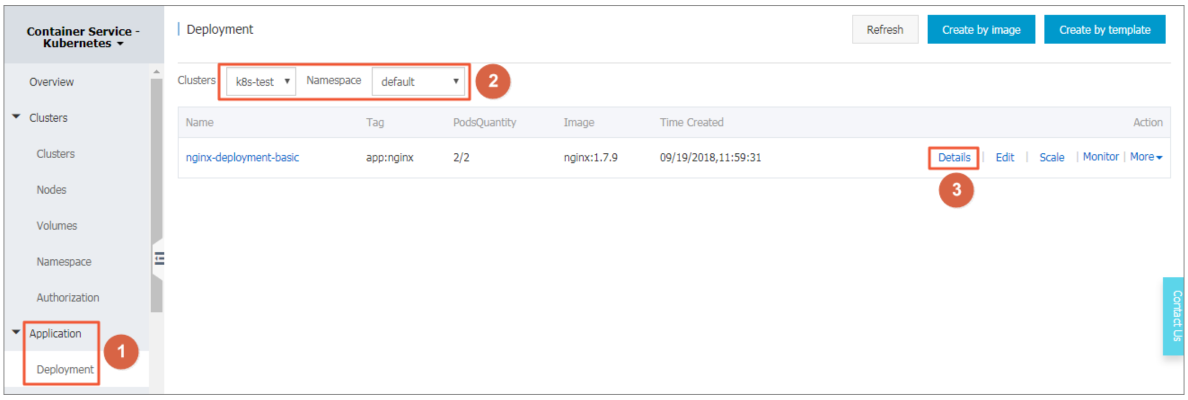
Alibaba Cloud Container Service Kubernetes supports the application trigger function. You can use an application trigger in many ways.

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

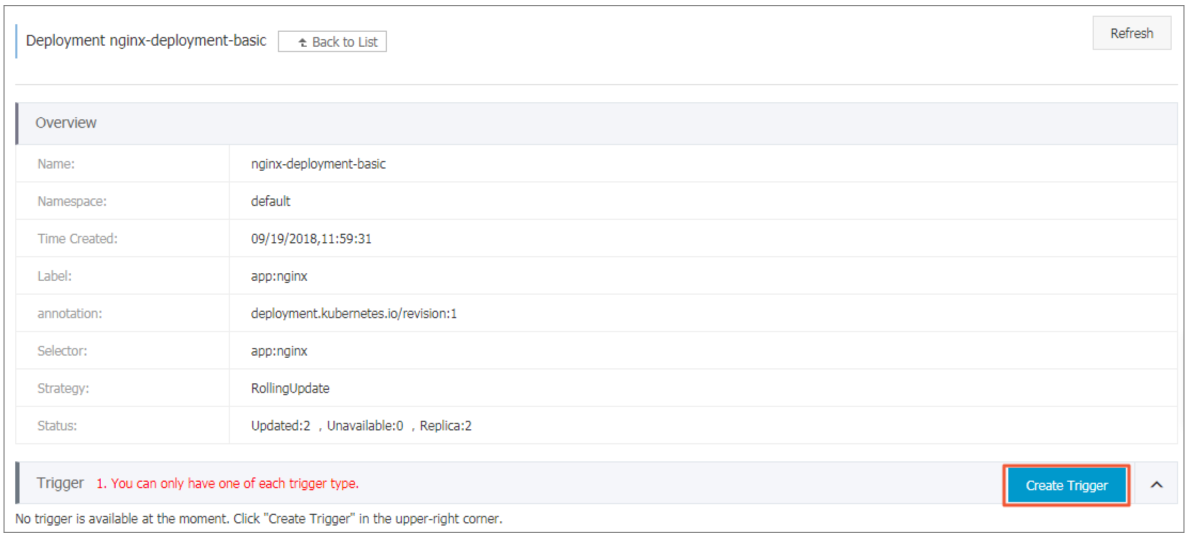
1. Log on to the *Container Service console*.
2. Click **Application** > **Deployment** and select a cluster and namespace. Click **Details** at the right of the target nginx application.



Clusters k8s-test Namespace default

Name	Tag	PodsQuantity	Image	Time Created	Action
nginx-deployment-basic	app:nginx	2/2	nginx:1.7.9	09/19/2018,11:59:31	Details | Edit | Scale | Monitor | More

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



Deployment nginx-deployment-basic   Back to List   Refresh

Overview

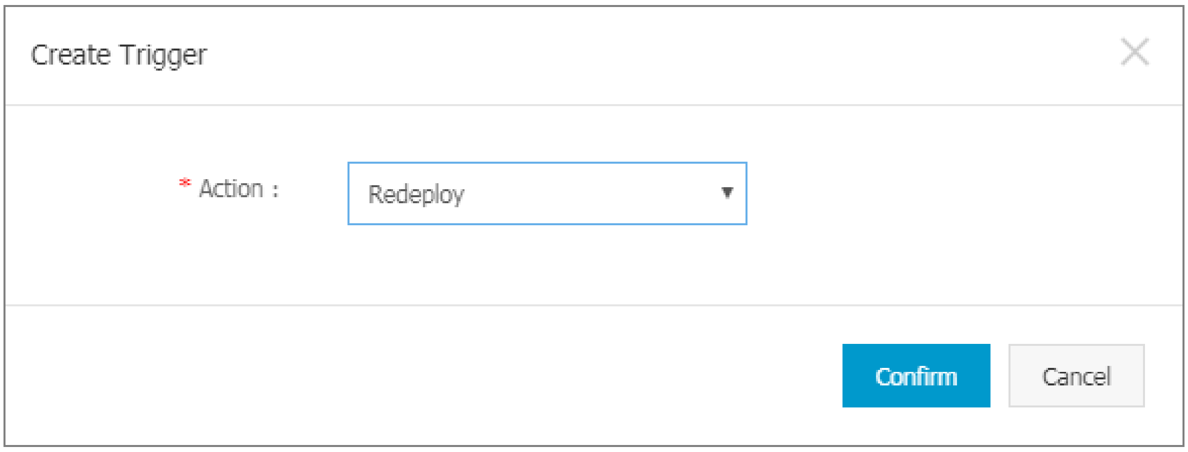
Name:	nginx-deployment-basic
Namespace:	default
Time Created:	09/19/2018,11:59:31
Label:	app:nginx
annotation:	deployment.kubernetes.io/revision:1
Selector:	app:nginx
Strategy:	RollingUpdate
Status:	Updated:2 , Unavailable:0 , Replica:2

Trigger 1. You can only have one of each trigger type. Create Trigger

No trigger is available at the moment. Click "Create Trigger" in the upper-right corner.

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

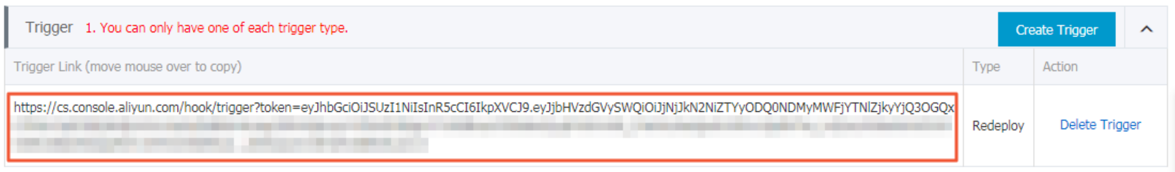
Note:
Currently, only the redeploy action is supported.



Create Trigger

* Action : Redeploy

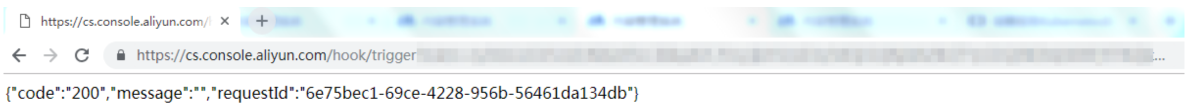
After the trigger is created, a trigger link is displayed in the trigger bar on the nginx application detail page.



Trigger 1. You can only have one of each trigger type. Create Trigger

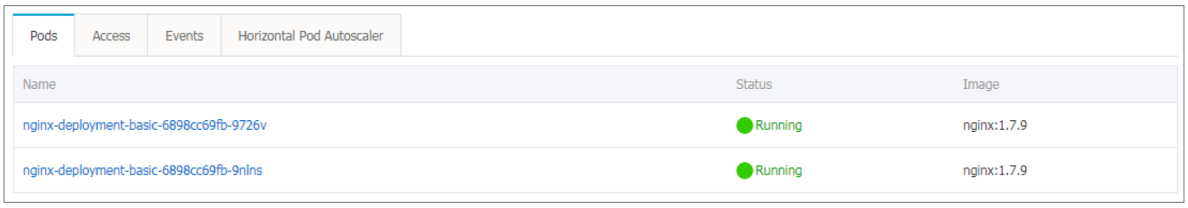
Trigger Link (move mouse over to copy)	Type	Action
https://cs.console.aliyun.com/hook/trigger?token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJjbHVzdGVySWQiOiJjNjJkN2NiZTYyODQ0NDMyMWFjYTNlZjkyYjQ3OGQx[redacted]	Redeploy	Delete Trigger

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.


https://cs.console.aliyun.com/hook/trigger

{"code":"200","message":"","requestId":"6e75bec1-69ce-4228-956b-56461da134db"}

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



Name	Status	Image
nginx-deployment-basic-6898cc69fb-9726v	● Running	nginx:1.7.9
nginx-deployment-basic-6898cc69fb-9nlns	● Running	nginx:1.7.9

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
```

1.6.13 View pods

You can view the pods of a Kubernetes cluster in the Container Service console or in the Kubernetes dashboard.

View pods in Container Service console

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Pods** in the left-side navigation pane to go to the Pods page.
3. Select the target cluster and namespace, the target pod, and click **Details** on the right.

Note:
You can update or delete a pod. For pods created by using deployments, we recommend that you manage these pods by using deployments.

4. View the pod details.

View pods in Kubernetes dashboard

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 **Pods** in the left-side navigation pane to view the pods in the cluster.

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

5. 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.

6. Click the pod name to view the details, CPU usage, and memory usage of the pod.

7. Click **LOGS** in the upper-right corner to view the pod logs.

8. You can also click the icon at the right of the pod and then select **Delete** to delete the pod.

1.6.14 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**.

1.6.15 Schedule a pod to a specified node

You can add a node label and then configure the `nodeSelector` 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

You have successfully created a Kubernetes cluster. For more information, see *#unique_31*.

Step 1 Add a node label

1. Log on to the *Container Service console*.
2. Under the Kubernetes menu, click **Clusters** > **Nodes** in the left-side navigation pane.
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. Under the Kubernetes menu, click **Applications** > **Deployment** in the left-side navigation pane.
3. Click **Create by template** in the upper-right corner.
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
       imagePullPolicy: IfNotPresent
       name: hello-pod
       ports:
         - containerPort: 8080
           protocol: TCP
       resources: {}
       securityContext:
         capabilities: {}
         privileged: false
       terminationMessagePath: /dev/termination-log
   dnsPolicy: ClusterFirst
   restartPolicy: Always
   nodeSelector:
     group: worker  ##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.

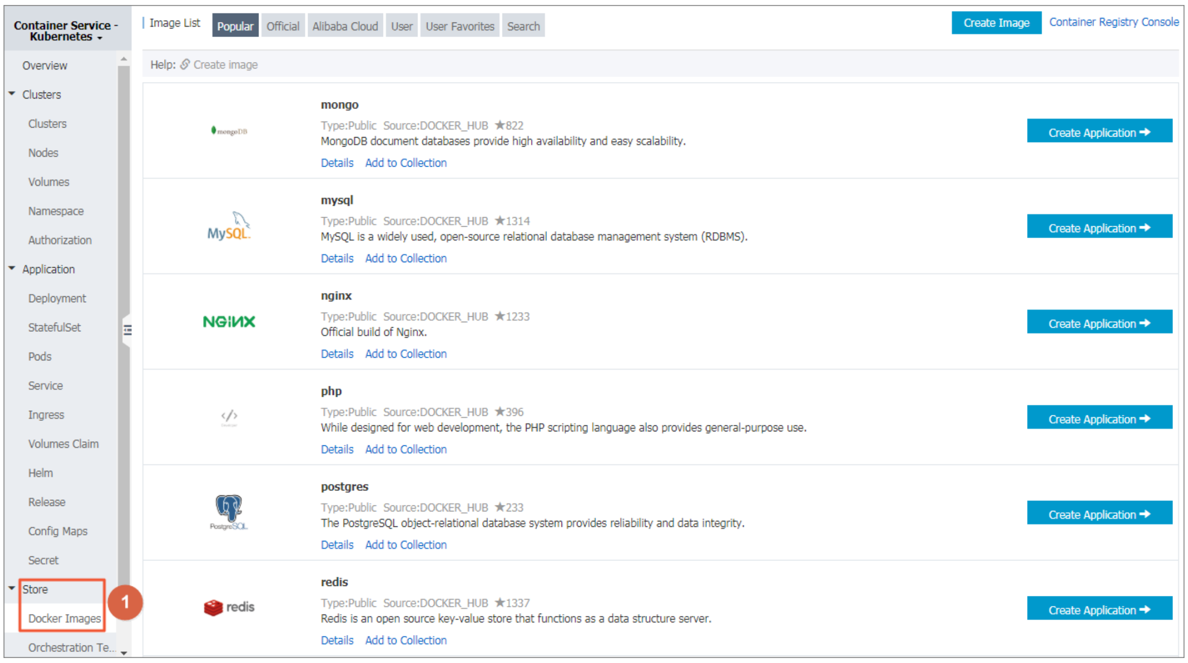
   You can view the information such as the pod label and node ID, which indicates the pod is successfully deployed to a node with the label `group:worker`.

1.6.16 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.



mongo
Type:Public Source:DOCKER_HUB ★822
MongoDB document databases provide high availability and easy scalability.

mysql
Type:Public Source:DOCKER_HUB ★1314
MySQL is a widely used, open-source relational database management system (RDBMS).

nginx
Type:Public Source:DOCKER_HUB ★1233
Official build of Nginx.

php
Type:Public Source:DOCKER_HUB ★396
While designed for web development, the PHP scripting language also provides general-purpose use.

postgres
Type:Public Source:DOCKER_HUB ★233
The PostgreSQL object-relational database system provides reliability and data integrity.

redis
Type:Public Source:DOCKER_HUB ★1337
Redis is an open source key-value store that functions as a data structure server.

You can view the image category.

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

1.6.17 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

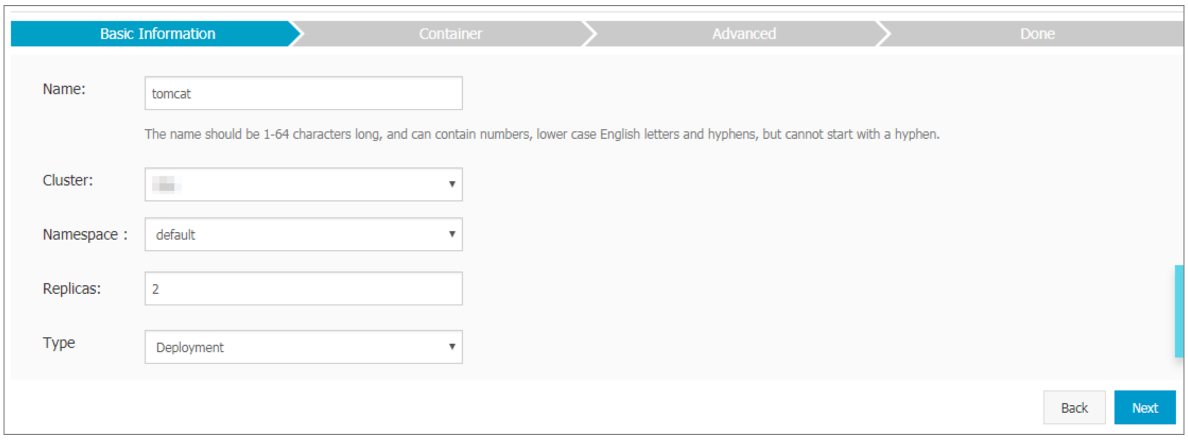
When you use a private image to create an application, you have to configure a secret for the image to secure the image. In the Container Service console, you can deliver the identity authentication information of the private image repository to Kubernetes through a secret of the docker-registry type.

Procedure

1. 登录##########。
2. Under the Kubernetes menu, click **Application** > **Deployment** in the left-side navigation pane, and 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.

   Name: tomcat
   The name should be 1-64 characters long, and can contain numbers, lower case English letters and hyphens, but cannot start with a hyphen.
   Cluster:
   Namespace : default
   Replicas: 2
   Type Deployment

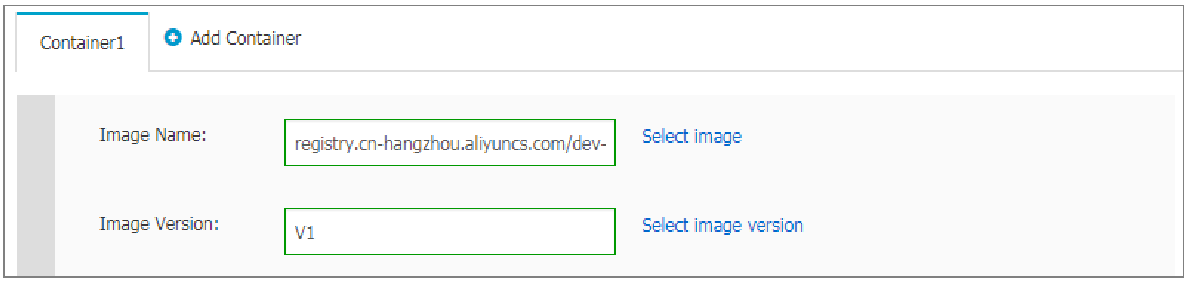
4. Configure containers.

   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.



registry.cn-hangzhou.aliyuncs.com/dev-

V1

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.

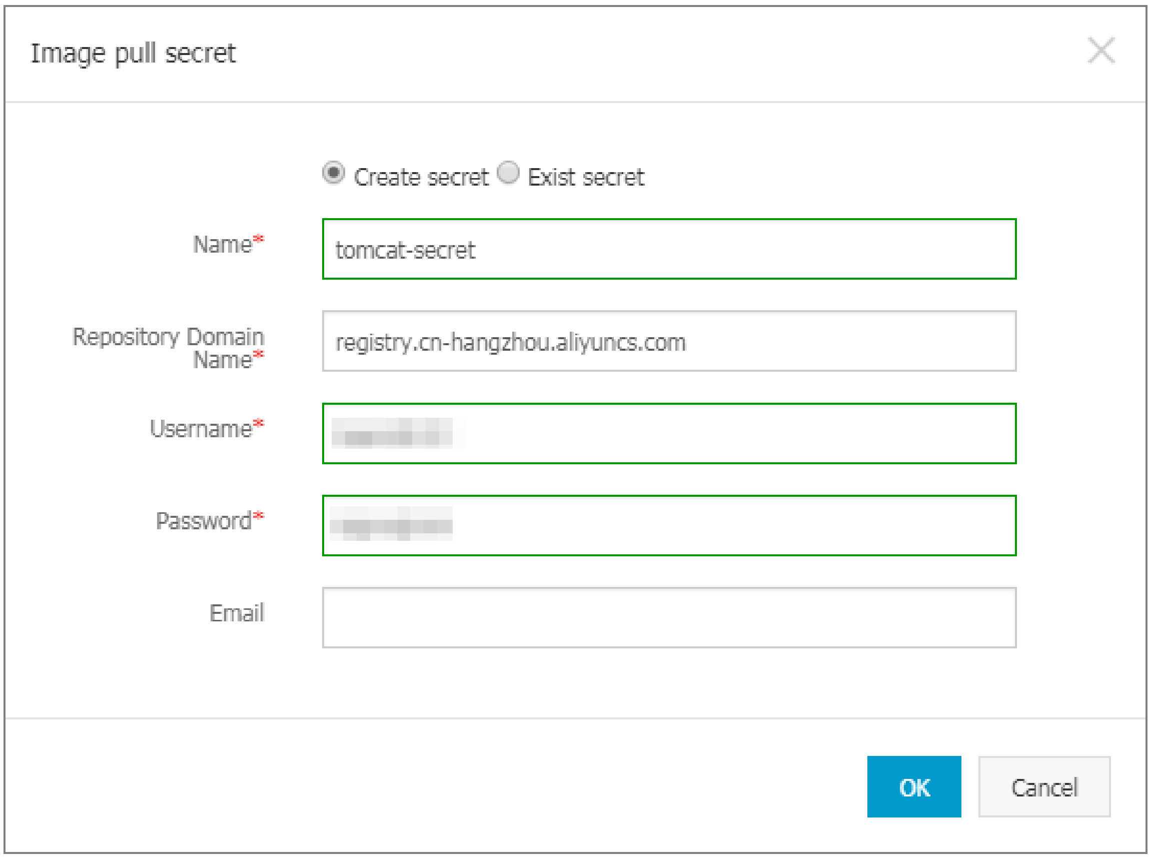


Image pull secret

[● Create secret] ○ Exist secret

Name*	tomcat-secret
Repository Domain Name*	registry.cn-hangzhou.aliyuncs.com
Username*	[redacted]
Password*	[redacted]
Email	

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

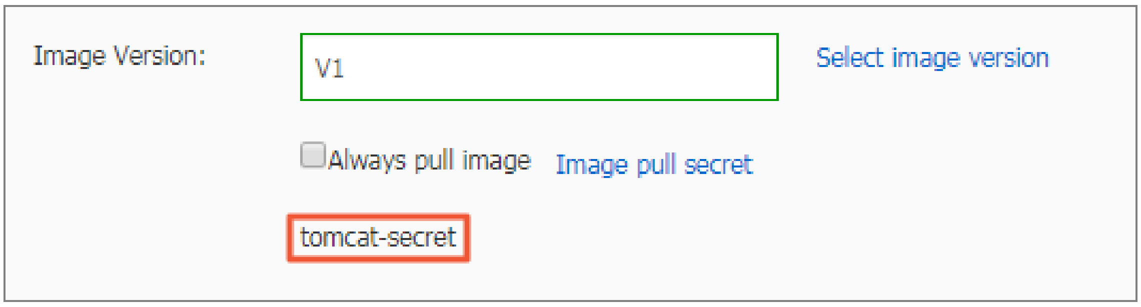
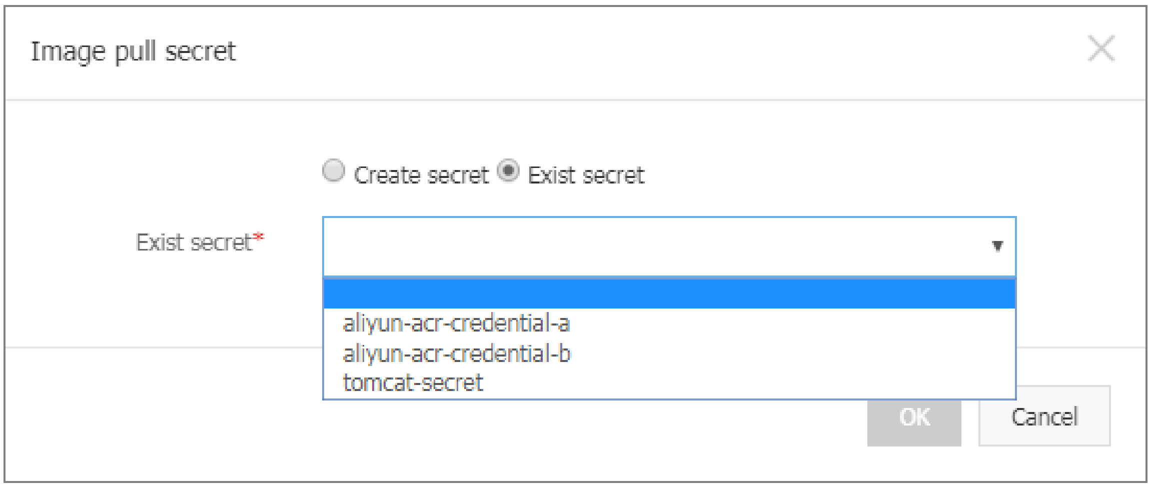


Image Version: V1

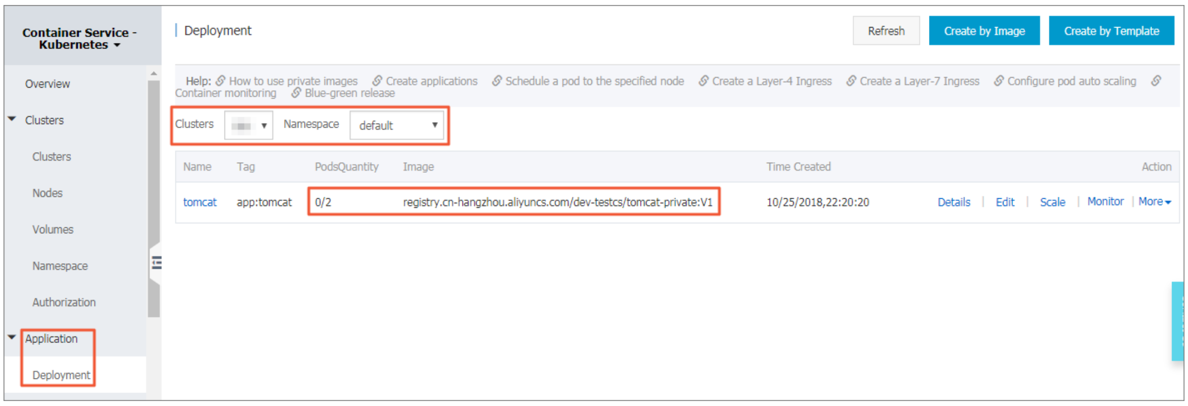
tomcat-secret

- 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 **Application** > **Deployment** 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.


Container Service - Kubernetes
Overview
Clusters
Clusters
Nodes
Volumes
Namespace
Authorization
Application
Deployment
Deployment
Refresh
Create by Image
Create by Template
Help: How to use private images · Create applications · Schedule a pod to the specified node · Create a Layer-4 Ingress · Create a Layer-7 Ingress · Configure pod auto scaling · Container monitoring · Blue-green release

Clusters [redacted] Namespace default

Name	Tag	PodsQuantity	Image	Time Created	Action
tomcat	app:tomcat	0/2	registry.cn-hangzhou.aliyuncs.com/dev-testcs/tomcat-private:V1	10/25/2018,22:20:20	Details | Edit | Scale | Monitor | More

1.7 Server Load Balancer and Ingress

1.7.1 Overview

Kubernetes clusters provide a diversity of approaches to access container applications, and support accessing internal services and realizing load balancing by means of Alibaba Cloud Server Load Balancer or Ingress.

1.7.2 Access services by using Server Load Balancer

You can access services by using Alibaba Cloud Server Load Balancer.

Note:

If cloud-controller-manager of your cluster is in v 1.9.3 or later versions, when you specify an existing SLB, the system does not process listeners for this SLB by default. You have to manually configure listeners for this SLB.

To view the version of cloud-controller-manager, execute 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
```

Operate by using command line

1. Create an Nginx application by using command line.

```
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 Alibaba Cloud Server Load Balancer 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=LoadBalancer
root@master # kubectl get svc
NAME                  CLUSTER-IP      EXTERNAL-IP      PORT(S)
                  AGE
nginx                 172.19.10.209   101.37.192.20   80:31891/TCP
                 4s
```

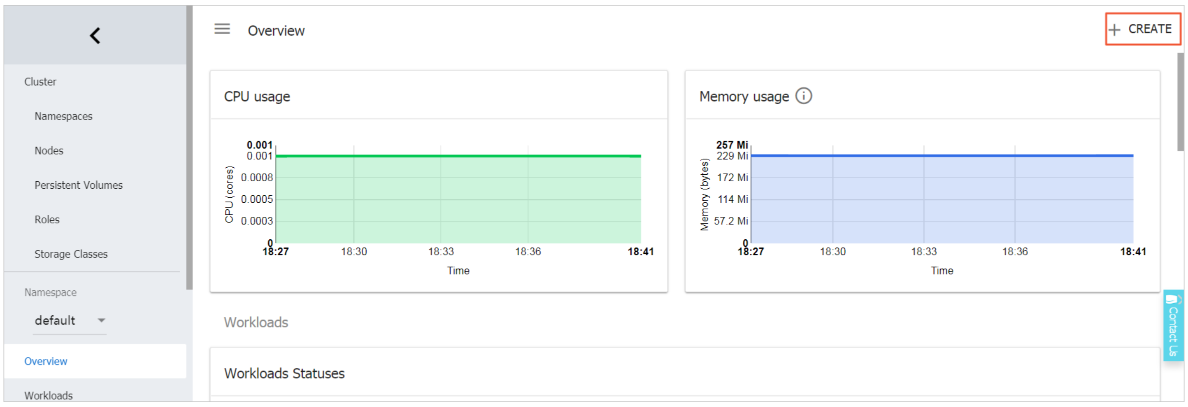
3. Visit `http://101.37.192.20` in the browser to access your Nginx service.

Operate by using 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: LoadBalancer
```

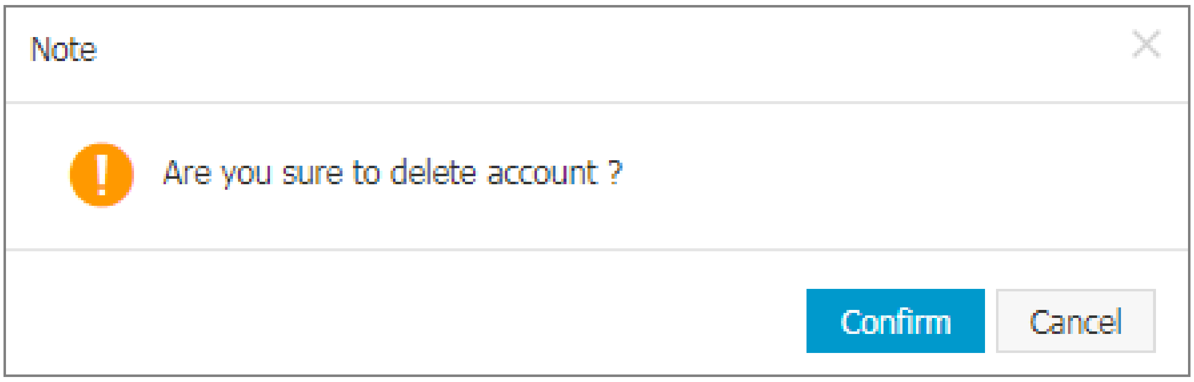
2. Log on to the *Container Service console*. Click **Dashboard** at the right of a cluster.
3. Click **CREATE** in the upper-right corner to create an application.



4. Click the **CREATE FROM FILE** tab. and then upload the *`nginx-svc.yml`* file you saved.
5. Click **UPLOAD**.

   An Alibaba Cloud Server Load Balancer instance that points to the created Nginx application is created. The service name is `http-svc`.
6. Select default under Namespace in the left-side navigation pane. Click **`Services`** in the left-side navigation pane.

   You can view the created Nginx service `http-svc` and the Server Load Balancer address `http://114.55.79.24:80`.



7. Copy the address to the browser to access the service.

More information

Alibaba Cloud Server Load Balancer also supports parameter configurations such as health check, billing method, and load balancing. For more information, see *Server Load Balancer configuration parameters*.

Annotations

Alibaba Cloud supports a plenty of Server Load Balancer features by using annotations.

Use existing intranet Server Load Balancer instance

You must specify two annotations. Replace with your own Server Load Balancer instance ID.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-address-type:
intranet
    service.beta.kubernetes.io/alicloud-loadbalancer-id: your-
loadbalancer-id
  labels:
    run: nginx
  name: nginx
  namespace: default
spec:
  ports:
  - name: web
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  sessionAffinity: None
  type: LoadBalancer
```

Save the preceding contents as slb.svc and then run the command `kubectl apply -f slb.svc`.

Create an HTTPS type Server Load Balancer instance

Create a certificate in the Alibaba Cloud console and record the cert-id. Then, use the following annotation to create an HTTPS type Server Load Balancer instance.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-cert-id: your-
cert-id
    service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port: "
https:443"
  labels:
    run: nginx
  name: nginx
  namespace: default
spec:
  ports:
  - name: web
    port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  sessionAffinity: None
  type: LoadBalancer
```

Note:

Annotations are case sensitive.

Annotation	Description	Default value
service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port	Use commas (,) to separate multiple values. For example, https:443,http:80.	None.
service.beta.kubernetes.io/alicloud-loadbalancer-address-type	The value is Internet or intranet.	Internet
service.beta.kubernetes.io/alicloud-loadbalancer-slb-network-type	Server Load Balancer network type. The value is classic or VPC.	classic
service.beta.kubernetes.io/alicloud-loadbalancer-charge-type	The value is paybytraffic or paybybandwidth.	paybybandwidth
service.beta.kubernetes.io/alicloud-loadbalancer-id	The Server Load Balancer instance ID. Specify an existing Server Load Balance	None.

Annotation	Description	Default value
	with the loadbalancer-id, and the existing listener is overwritten. Server Load Balancer is not deleted when the service is deleted.	
service.beta.kubernetes.io/alicloud-loadbalancer-backend-label	Use label to specify which nodes are mounted to the Server Load Balancer backend.	None.
service.beta.kubernetes.io/alicloud-loadbalancer-region	The region in which Server Load Balancer resides.	None.
service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth	Server Load Balancer bandwidth.	50
service.beta.kubernetes.io/alicloud-loadbalancer-cert-id	ID of a certificate on Alibaba Cloud. You must have uploaded a certificate first.	""
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-flag	The value is on or off.	The default value is off. No need to modify the TCP parameters because TCP enables health check by default, which cannot be configured.
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-type	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-uri	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-port	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-healthy-threshold	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-unhealthy-threshold	See *HealthCheck*.	

Annotation	Description	Default value
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout	See *HealthCheck*.	

1.7.3 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-port80
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
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /svc
        backend:
          serviceName: http-svc
          servicePort: 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:
          serviceName: http-svc1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: http-svc2
          servicePort: 80
  - host: foo.example.com
    http:
      paths:
      - path: /film
        backend:
          serviceName: http-svc3
          servicePort: 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].alicontainer.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].alicontainer.com ##Replace with
 the default service access domain name of your cluster.
    http:
      paths:
      - path: /
        backend:
          serviceName: http-svc1
          servicePort: 80
  - host: bar.[cluster-id].[region-id].alicontainer.com ##Replace with
 the default service access domain name of your cluster.
    http:
      paths:
      - path: /
        backend:
          serviceName: http-svc2
          servicePort: 80
EOF
root@master # kubectl get ing
NAME HOSTS ADDRESS PORTS AGE
shared-dns foo.[cluster-id].[region-id].alicontainer.com,bar.[cluster-
id].[region-id].alicontainer.com 47.95.160.171 80 40m
```

Then, you can access the `http-svc1` service by using `http://foo.[cluster-id].[region-id].alicontainer.com/`and access the `http-svc2` service by using `http://bar.[cluster-id].[region-id].alicontainer.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. 1. 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:
             serviceName: http-svc1
             servicePort: 80
         - path: /bar
           backend:
             serviceName: http-svc2
             servicePort: 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`.

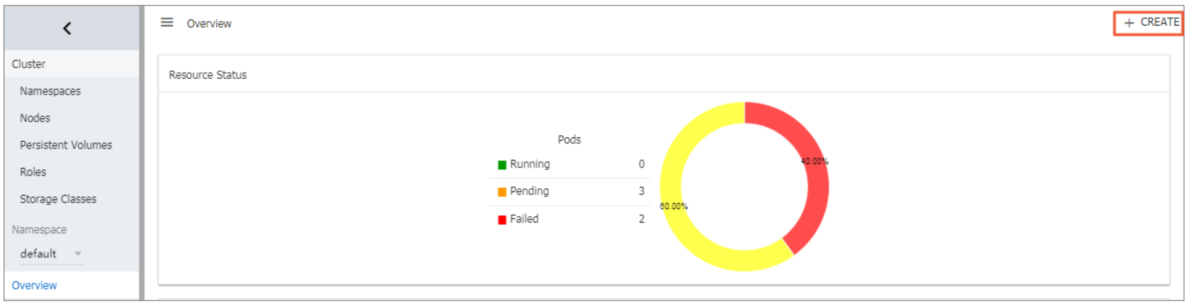
You can also access the HTTPS service by using HTTP. By default, Ingress redirects HTTP access configured with HTTPS to the HTTPS address. Therefore, access to `http://foo.bar.com/foo` will be automatically redirected to `https://foo.bar.com/foo`.

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:
          serviceName: http-svc
          servicePort: 80
```

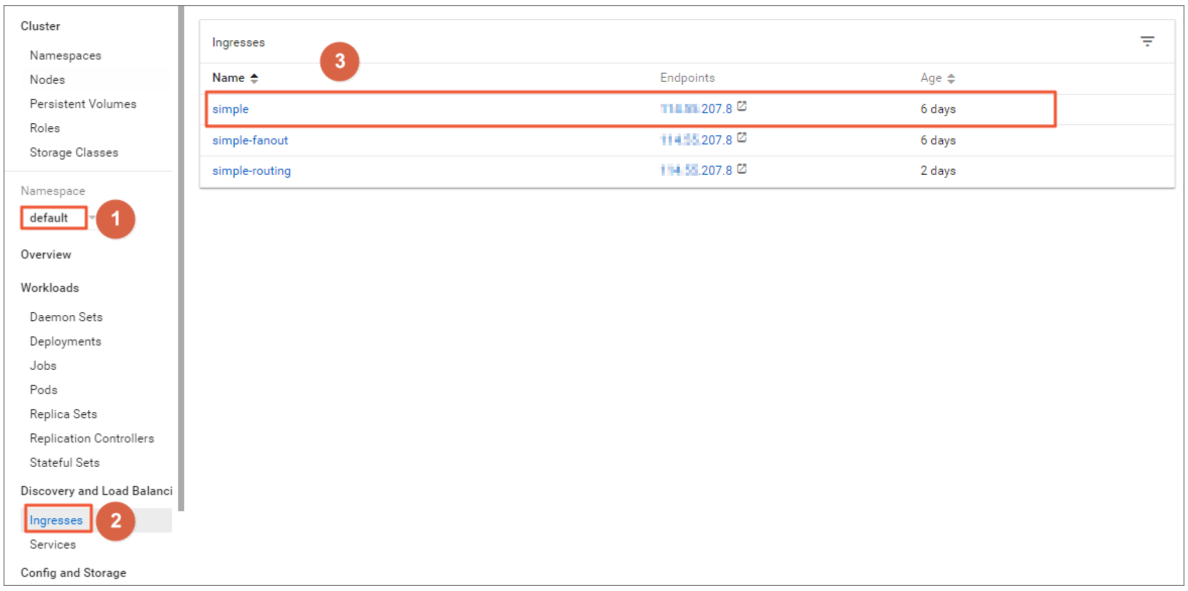
2. Log on to the ##########. Under Kubernetes, click Clusters in the left-side navigation pane. Click **Dashboard**at the right of the cluster to enter the Kubernetes dashboard.
3. Click **CREATE** in the upper-right corner to create an application.



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`.



Name	Endpoints	Age
simple	[redacted].207.8	6 days
simple-fanout	[redacted].207.8	6 days
simple-routing	[redacted].207.8	2 days

7. Enter the address in the browser to access the created `http-svc` service.

1.7.4 Configure Ingress monitoring

You can view the Ingress monitoring data by enabling the default VTS module of Ingress.

Enable VTS module by running commands

1. Modify the Ingress ConfigMap configuration to add the configuration item `enable-vts-status: "true"`.

```
root@master # kubectl edit configmap nginx-configuration -n kube-
system
configmap "nginx-configuration" edited
```

After the modification, the contents of the Ingress ConfigMap are as follows:

```
apiVersion: v1
data:
  enable-vts-status: "true" # Enable VTS module
  proxy-body-size: 20m
kind: ConfigMap
metadata:
  Annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"proxy-body-size":"20m"},"kind":"
ConfigMap","metadata":{"annotations":{},"labels":{"app":"ingress-
nginx"},"name":"nginx-configuration","namespace":"kube-system"}}
  creationTimestamp: 2018-03-20T07:10:18Z
  labels:
    app: ingress-nginx
  name: nginx-configuration
  namespace: kube-system
  selfLink: /api/v1/namespaces/kube-system/configmaps/nginx-
configuration
```

2. Verify if Ingress Nginx has enabled the VTS module normally.

```
root@master # kubectl get pods --selector=app=ingress-nginx -n kube-
system
NAME READY STATUS RESTARTS AGE
nginx-ingress-controller-79877595c8-78gq8 1/1 Running 0 1h
root@master # kubectl exec -it nginx-ingress-controller-79877595c8-
78gq8 -n kube-system -- cat /etc/nginx/nginx.conf | grep vhost_traf
fic_status_display
          vhost_traffic_status_display;
          vhost_traffic_status_display_format html;
```

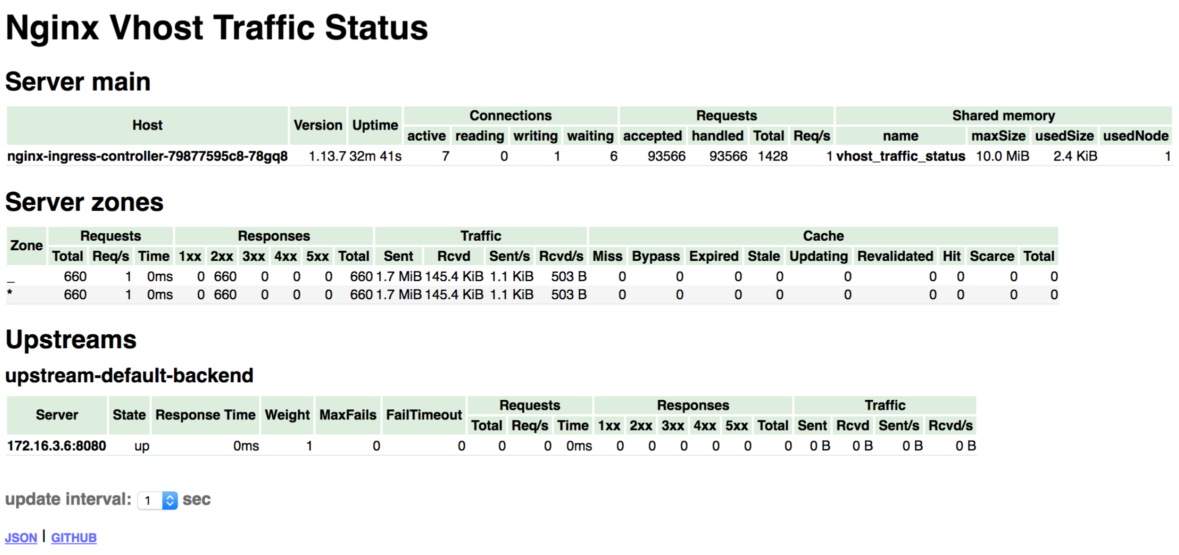
3. Locally access the Ingress Nginx monitoring console.

Note:

By default, the VTS port is not opened for security considerations. Here use the port-forward method to access the console.

```
root@master # kubectl port-forward nginx-ingress-controller-
79877595c8-78gq8 -n kube-system 18080
Forwarding from 127.0.0.1:18080 -> 18080
Handling connection for 18080
```

4. Use `http://localhost:18080/nginx_status` to access the VTS monitoring console.



Nginx Vhost Traffic Status

Server main

Host	Version	Uptime	Connections				Requests				Shared memory			
			active	reading	writing	waiting	accepted	handled	Total	Req/s	name	maxSize	usedSize	usedNode
nginx-ingress-controller-79877595c8-78gq8	1.13.7	32m 41s	7	0	1	6	93566	93566	1428	1	vhost_traffic_status	10.0 MiB	2.4 KiB	1

Server zones

Zone	Requests			Responses						Traffic				Cache								
	Total	Req/s	Time	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s	Miss	Bypass	Expired	Stale	Updating	Revalidated	Hit	Scarce	Total
_	660	1	0ms	0	660	0	0	0	660	1.7 MiB	145.4 KiB	1.1 KiB	503 B	0	0	0	0	0	0	0	0	0
*	660	1	0ms	0	660	0	0	0	660	1.7 MiB	145.4 KiB	1.1 KiB	503 B	0	0	0	0	0	0	0	0	0

Upstreams

upstream-default-backend

Server	State	Response Time	Weight	MaxFails	FailTimeout	Requests			Responses						Traffic			
						Total	Req/s	Time	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s
172.16.3.6:8080	up	0ms	1	0	0	0	0	0ms	0	0	0	0	0	0	0 B	0 B	0 B	0 B

update interval: 1 sec

JSON | GITHUB

Enable VTS module by using the Kubernetes dashboard

1. Log on to the *Container Service console*.
2. On the Cluster List page of Kubernetes clusters, click **Dashboard** at the right of a cluster to enter the Kubernetes dashboard page.

3. Select kube-system under Namespace in the left-side navigation pane. Click Config Maps in the left-side navigation pane. Click the icon at the right of nginx-configuration and then select View/edit YAML. Edit the config map to add the configuration item `enable-vts-status: "true"`.

   The contents of the saved Ingress ConfigMap are as follows:

```
{
  "kind": "ConfigMap",
  "apiVersion": "v1",
  "metadata": {
    "name": "nginx-configuration",
    "namespace": "kube-system",
    "selfLink": "/api/v1/namespaces/kube-system/configmaps/nginx-
configuration",
    "creationTimestamp": "2018-03-20T07:10:18Z",
    "labels": {
      "app": "ingress-nginx"
    },
    "annotations": {
      "kubectl.kubernetes.io/last-applied-configuration": "{\"
apiVersion\":\"v1\",\"data\":{\"proxy-body-size\":\"20m\"},\"kind\":
\"ConfigMap\",\"metadata\":{\"annotations\":{},\"labels\":{\"app\":
\"ingress-nginx\"},\"name\":\"nginx-configuration\",\"namespace\":\"
kube-system\"}}\n"
   }
 },
 "data": {
   "proxy-body-size": "20m",
   "enable-vts-status": "true"
 }
}
```

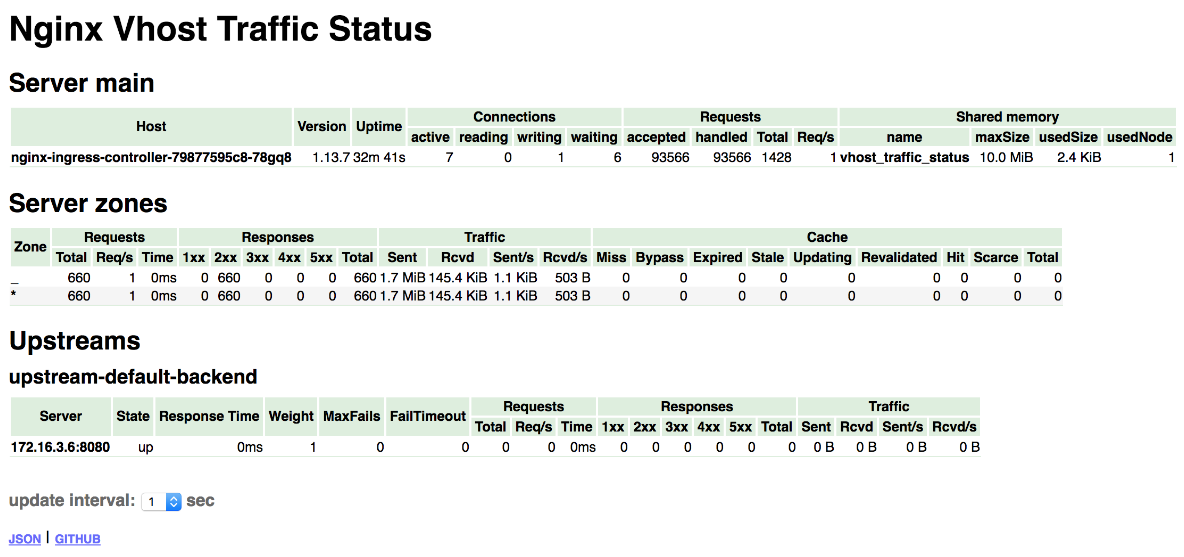
4. Locally access the Ingress Nginx monitoring console.

   Note:

   By default, the VTS port is not opened for security considerations. Here use the port-forward method to access the console.

```
root@master # kubectl port-forward nginx-ingress-controller-
79877595c8-78gq8 -n kube-system 18080
Forwarding from 127.0.0.1:18080 -> 18080
Handling connection for 18080
```

5. Use `http://localhost:18080/nginx_status` to access the VTS monitoring console.



Nginx Vhost Traffic Status

Server main

Host	Version	Uptime	Connections				Requests				Shared memory			
			active	reading	writing	waiting	accepted	handled	Total	Req/s	name	maxSize	usedSize	usedNode
nginx-ingress-controller-79877595c8-78gq8	1.13.7	32m 41s	7	0	1	6	93566	93566	1428	1	vhost_traffic_status	10.0 MiB	2.4 KiB	1

Server zones

Zone	Requests			Responses						Traffic				Cache								
	Total	Req/s	Time	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s	Miss	Bypass	Expired	Stale	Updating	Revalidated	Hit	Scarce	Total
_	660	1	0ms	0	660	0	0	0	660	1.7 MiB	145.4 KiB	1.1 KiB	503 B	0	0	0	0	0	0	0	0	0
*	660	1	0ms	0	660	0	0	0	660	1.7 MiB	145.4 KiB	1.1 KiB	503 B	0	0	0	0	0	0	0	0	0

Upstreams

upstream-default-backend

Server	State	Response Time	Weight	MaxFails	FailTimeout	Requests			Responses						Traffic			
						Total	Req/s	Time	1xx	2xx	3xx	4xx	5xx	Total	Sent	Rcvd	Sent/s	Rcvd/s
172.16.3.6:8080	up	0ms	1	0	0	0	0	0ms	0	0	0	0	0	0	0 B	0 B	0 B	0 B

update interval: 1 sec

JSON | GITHUB

1.7.5 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 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:
  Annotations:
    nginx.ingress.kubernetes.io/service-match: 'new-nginx: header("foo
", /^bar$/)' #Grayscale publish rule, this example is request header
    Nginx. ingress. kubernetes. IO/service-weight: 'New-nginx: 50, old
-nginx: 50' # FTraffic weight annotation
  creationTimestamp: null
  generation: 1
  name: nginx-ingress
  selfLink: /apis/extensions/v1beta1/namespaces/default/ingresses/
nginx-ingress
spec:
  rules: ##The Ingress rule
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: new-nginx
          servicePort: 80
          path: /
        - backend:
            serviceName: old-nginx
            servicePort: 80
          path: /
tls: ## Enable TLS to configure the secure Ingress service
  - hosts:
    - *.xxxxxx.cn-hangzhou.alicontainer.com
    - foo.bar.com
    secretName: nginx-ingress-secret ##The name of the used secret.
status:
  loadBalancer: {}
```

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.

1.7.6 Create an Ingress in Container Service console

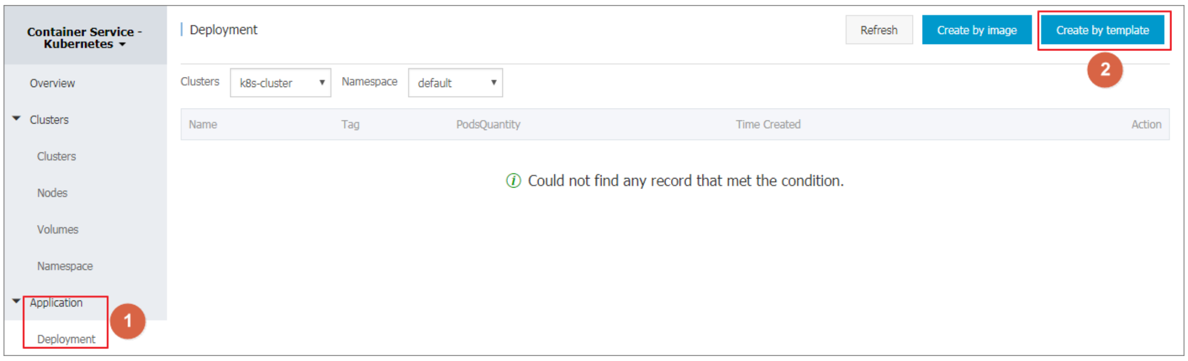
Alibaba Cloud Container Service console integrates with the Ingress service, which allows you to quickly create an Ingress service in the Container Service console to build the flexible and reliable traffic access layer.

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

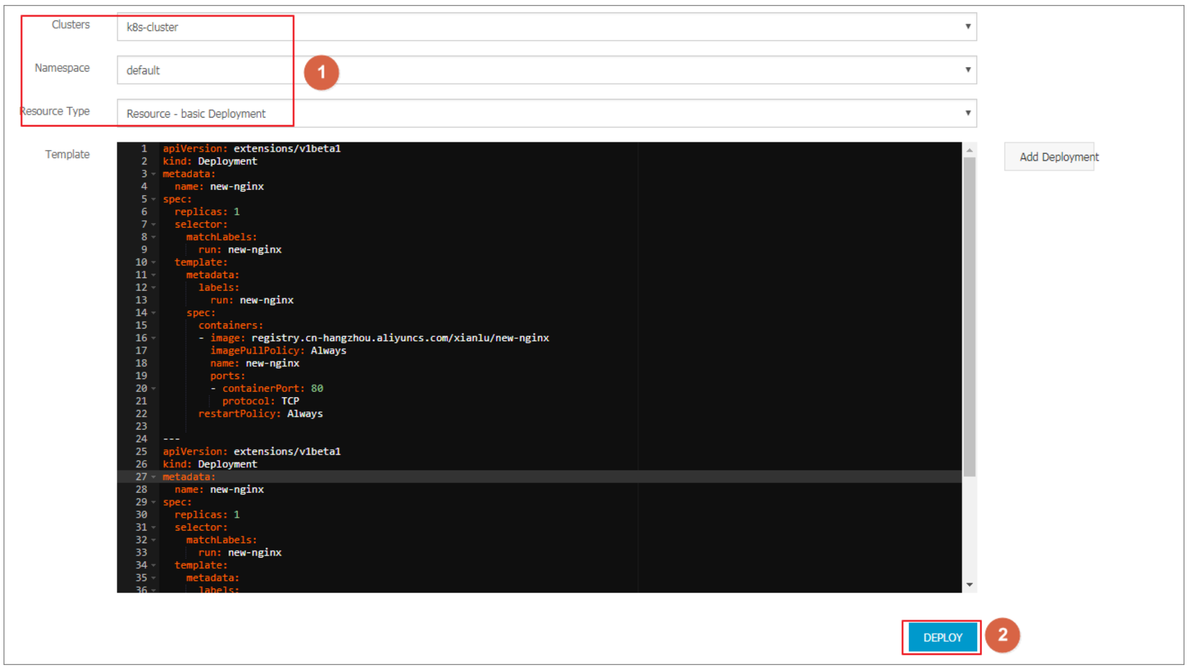
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane to enter the Deployment List page.
3. Click **Create by template** in the upper-right corner.



Could not find any record that met the condition.

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).



```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: new-nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      run: new-nginx
  template:
    metadata:
      labels:
        run: new-nginx
    spec:
      containers:
      - image: registry.cn-hangzhou.aliyuncs.com/xianlu/new-nginx
        imagePullPolicy: Always
        name: new-nginx
        ports:
        - containerPort: 80
          protocol: TCP
      restartPolicy: Always

---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: new-nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      run: new-nginx
  template:
    metadata:
      labels:
```

The orchestration template for old-nginx is as follows:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: old-nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      run: old-nginx
  template:
    metadata:
      labels:
        run: old-nginx
    spec:
      containers:
      - image: registry.cn-hangzhou.aliyuncs.com/xianlu/old-nginx
        imagePullPolicy: Always
        name: old-nginx
        ports:
        - containerPort: 80
          protocol: TCP
      restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
  name: old-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: old-nginx
  sessionAffinity: None
  type: NodePort
```

The orchestration template for new-nginx is as follows:

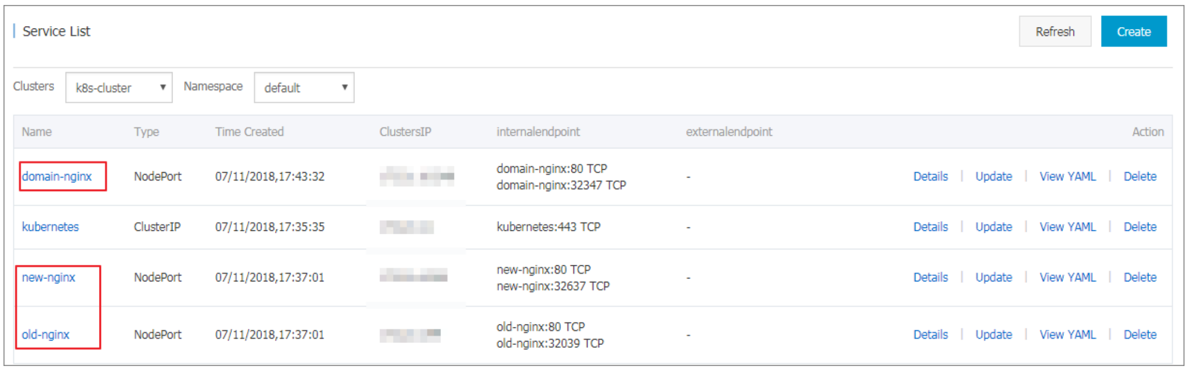
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: new-nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      run: new-nginx
  template:
    metadata:
      labels:
        run: new-nginx
    spec:
      containers:
      - image: registry.cn-hangzhou.aliyuncs.com/xianlu/new-nginx
        imagePullPolicy: Always
        name: new-nginx
        ports:
        - containerPort: 80
          protocol: TCP
      restartPolicy: Always
---
apiVersion: v1
kind: Service
metadata:
  name: new-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: new-nginx
  sessionAffinity: None
  type: NodePort
```

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:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <
image_name:tags>
        ports:
        - containerPort: 80

---
apiVersion: v1
kind: Service
metadata:
  name: domain-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  sessionAffinity: None
  type: NodePort
```

5. Click **Application** > **Service** in the left-side navigation pane to enter the Services List page.

   After the service is created, you can see it on the Service List page.



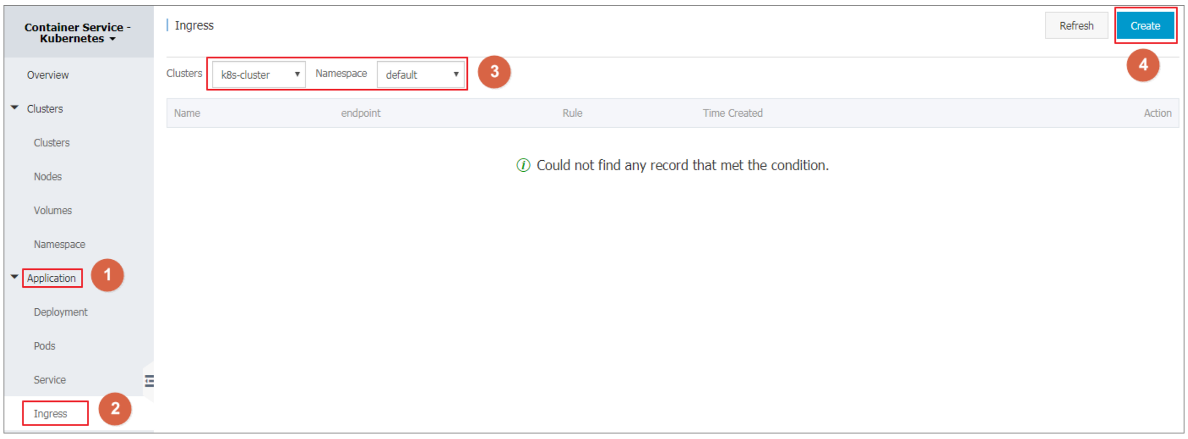
Service List

Clusters: k8s-cluster   Namespace: default

Name	Type	Time Created	ClustersIP	internalendpoint	externalendpoint	Action
domain-nginx	NodePort	07/11/2018,17:43:32	[redacted]	domain-nginx:80 TCP domain-nginx:32347 TCP	-	Details | Update | View YAML | Delete
kubernetes	ClusterIP	07/11/2018,17:35:35	[redacted]	kubernetes:443 TCP	-	Details | Update | View YAML | Delete
new-nginx	NodePort	07/11/2018,17:37:01	[redacted]	new-nginx:80 TCP new-nginx:32637 TCP	-	Details | Update | View YAML | Delete
old-nginx	NodePort	07/11/2018,17:37:01	[redacted]	old-nginx:80 TCP old-nginx:32039 TCP	-	Details | Update | View YAML | Delete

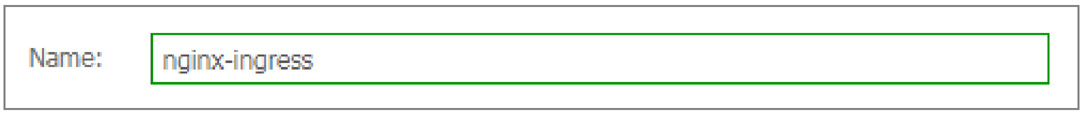
Step 2. Create an Ingress

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. Then click **Create** in the upper-right corner.



Could not find any record that met the condition.

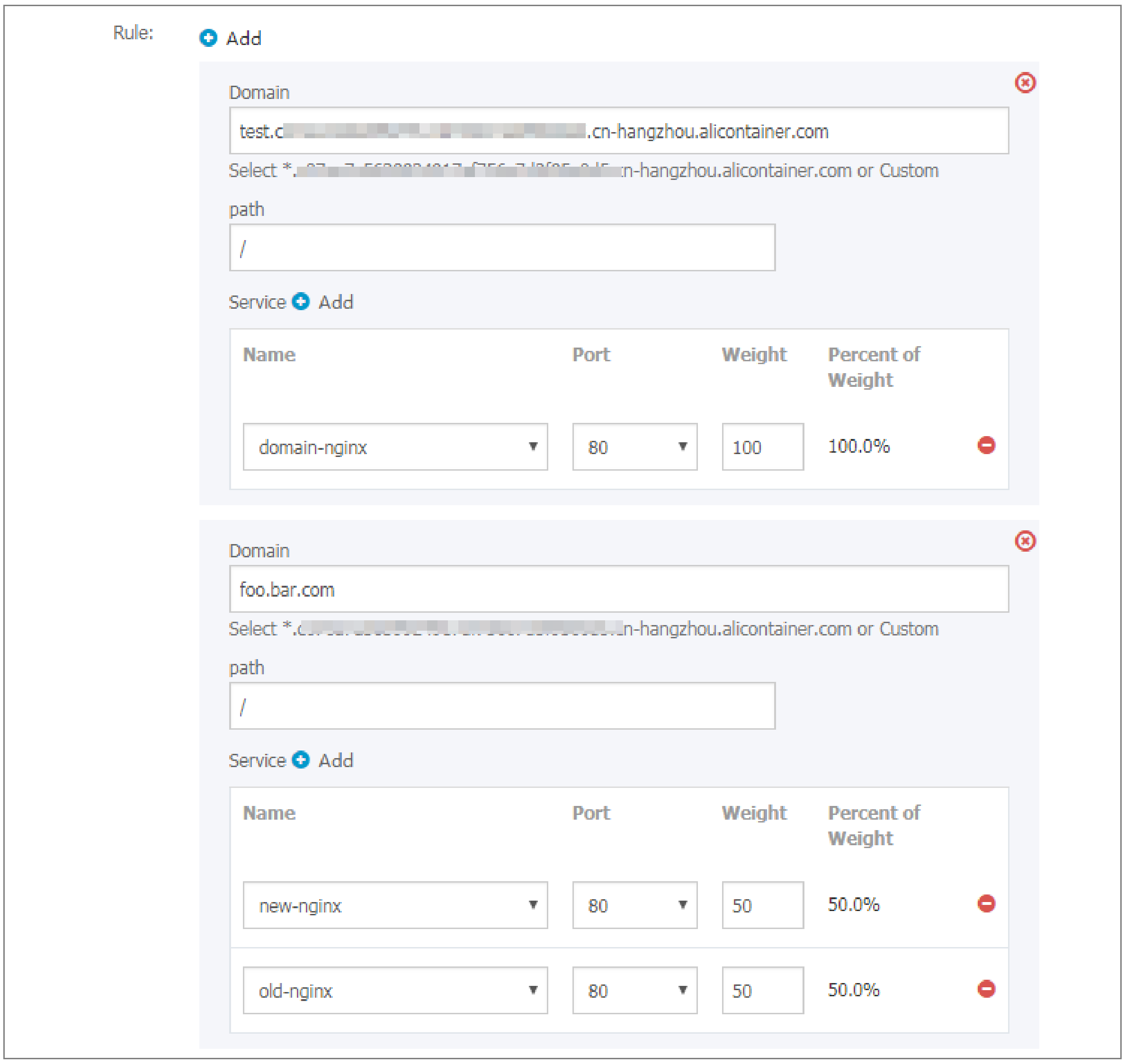
4. In the displayed dialog box, enter the Ingress name. In this example, enter nginx-ingress.



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.



Rule: Add

Domain

test.c[redacted].cn-hangzhou.alicontainer.com

Select *.[redacted]n-hangzhou.alicontainer.com or Custom

path

/

Service Add

Name	Port	Weight	Percent of Weight
domain-nginx	80	100	100.0%

Domain

foo.bar.com

Select *.c[redacted]n-hangzhou.alicontainer.com or Custom

path

/

Service Add

Name	Port	Weight	Percent of Weight
new-nginx	80	50	50.0%
old-nginx	80	50	50.0%

- 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].alicontainer.com`.

    The default domain name of this cluster is displayed in the Create dialog box, in the `*.[cluster-id].[region-id].alicontainer.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.

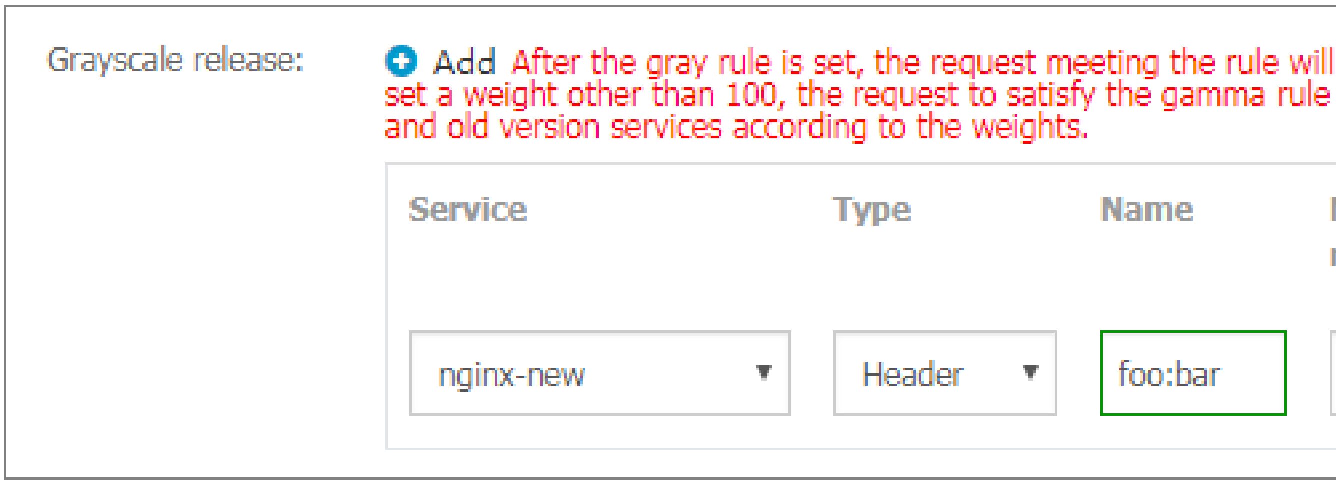
   Note:
   Currently, the Alibaba Cloud Container Service Kubernetes Ingress Controller requires **`0.12.0-5`** and above to support the traffic segmentation feature.

   Container Service supports different traffic segmentation methods for grayscale publish and AB test scenarios.

   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 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 correspond ing 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.



After the gray rule is set, the request meeting the rule will
set a weight other than 100, the request to satisfy the gamma rule
and old version services according to the weights.

Service	Type	Name
nginx-new	Header	foo:bar

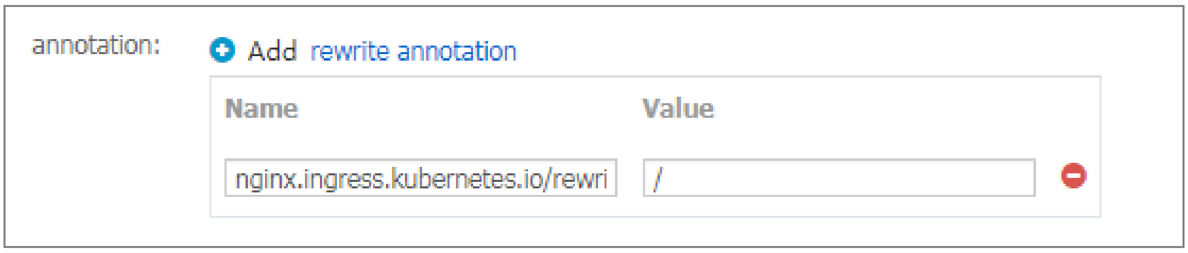
- **Service**: Routing rule configuration service.
- **Type**: matching request header, cookie, and query (request) parameters are supported.
- **Name and match value**: User-defined request field, name and match value are key-value pairs.
- **Match rules**: Regular and exact matches are supported.

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/*.

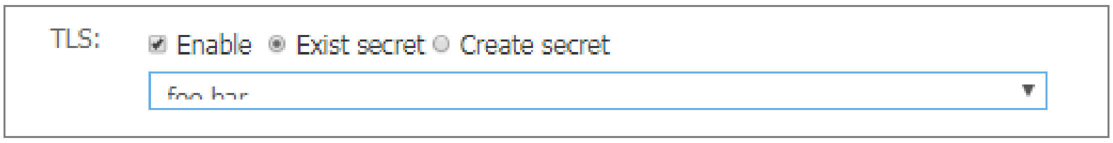


annotation: Add rewrite annotation

Name	Value
nginx.ingress.kubernetes.io/rewri	/

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.



TLS: [✓ Enable] [✓ Exist secret] Create secret

foo.bar

1. Log on to the master node and create *tls.key* and *tls.crt*.

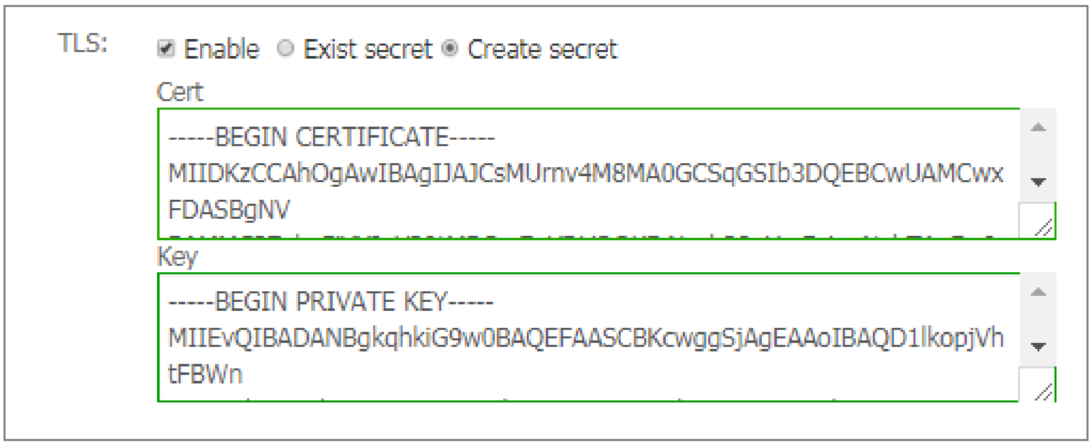
```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls
.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
```

2. Create a secret.

```
kubectl create secret tls foo.bar --key tls.key --cert tls.crt
```

3. 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.



TLS: [✓ Enable] Exist secret [✓ Create secret]

Cert

```
-----BEGIN CERTIFICATE-----
MIIDKzCCAhOgAwIBAgIJAJCsMUrnv4M8MA0GCSqGSIb3DQEBCwUAMCwx
FDASBgNV
```

Key

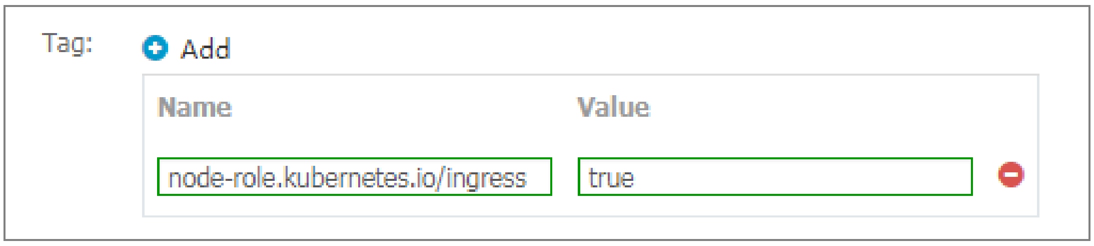
```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQD1lkopjVh
tFBWn
```

1. 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"
   ```
2. Run the `vim tls.key` and `vim tls.crt` to get the generated private key and certificate.
3. Copy the generated certificate and private key to the Cert and Key fields.

9. Adding the tags.

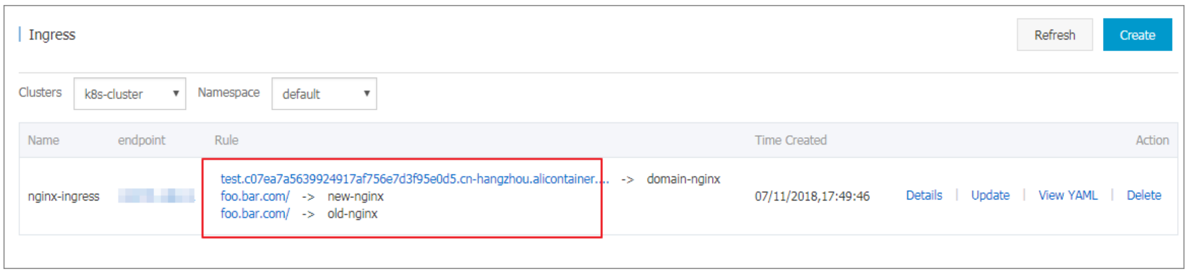
   Add the corresponding tags for Ingress to indicate the characteristics of the Ingress.



Name	Value
node-role.kubernetes.io/ingress	true

10. Click **Create**.

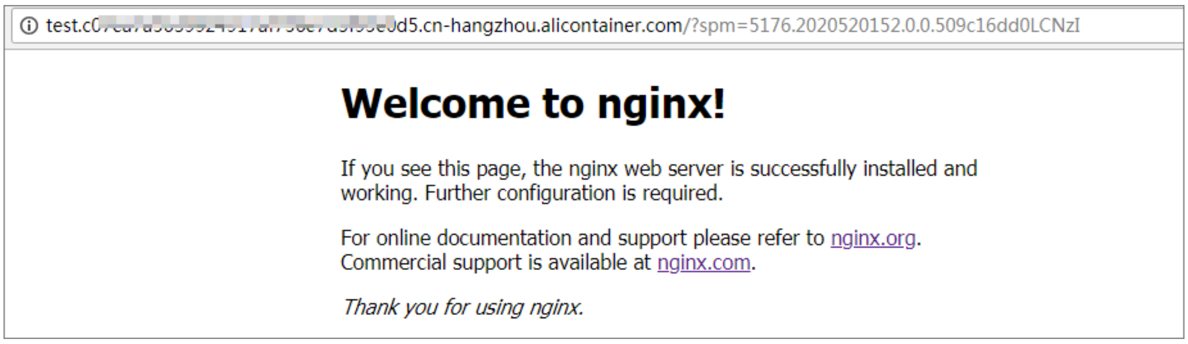
    The Ingress nginx-ingress is displayed on the Ingress page.



Clusters k8s-cluster Namespace default

Name	endpoint	Rule	Time Created	Action
nginx-ingress	[redacted]	test.c07ea7a5639924917af756e7d3f95e0d5.cn-hangzhou.alicontainer... -> domain-nginx foo.bar.com/ -> new-nginx foo.bar.com/ -> old-nginx	07/11/2018,17:49:46	Details | Update | View YAML | Delete

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.



test.c0[redacted]d5.cn-hangzhou.alicontainer.com/?spm=5176.2020520152.0.0.509c16dd0LCNzI

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

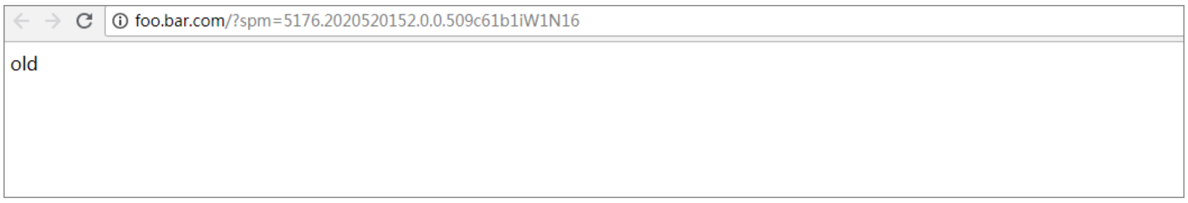
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using 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.



old

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
```

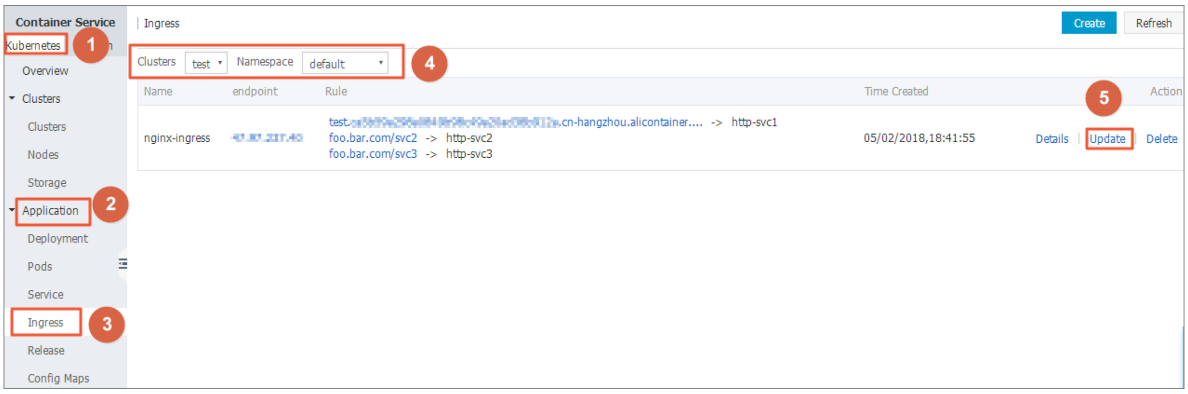
1.7.7 Update an Ingress

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 Container Service console*.

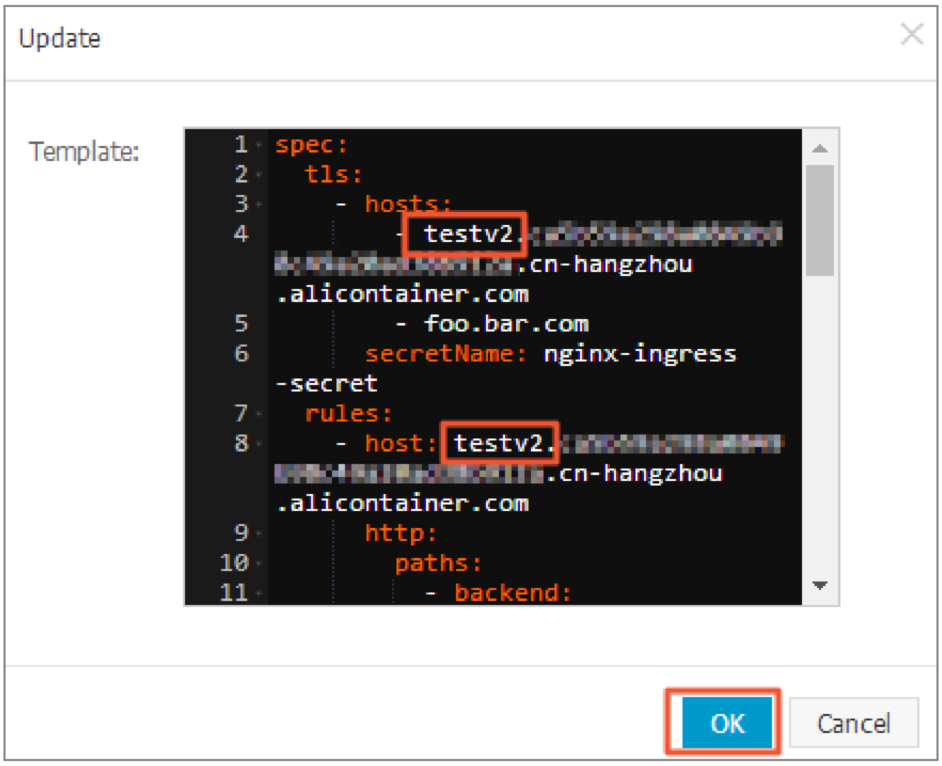
Procedure

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.



Name	endpoint	Rule	Time Created	Action
nginx-ingress	[redacted]	test.[redacted].cn-hangzhou.alicontainer.... -> http-svc1 foo.bar.com/svc2 -> http-svc2 foo.bar.com/svc3 -> http-svc3	05/02/2018,18:41:55	Details | Update | Delete

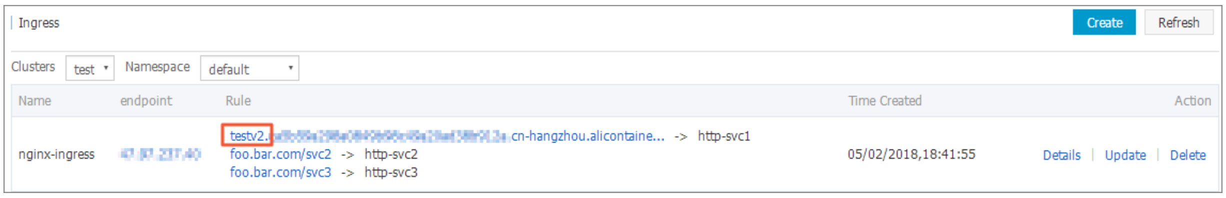
4. Update the Ingress parameters in the displayed dialog box and then click **OK**. change `test.[cluster-id].[region-id].alicontainer.com` to `testv2.[cluster-id].[region-id].alicontainer.com`。



```
spec:
  tls:
    - hosts:
        - testv2.[redacted].cn-hangzhou.alicontainer.com
        - foo.bar.com
      secretName: nginx-ingress-secret
  rules:
    - host: testv2.[redacted].cn-hangzhou.alicontainer.com
      http:
        paths:
          - backend:
```

What's next

On the Ingress page, you can see a rule of this Ingress is changed.



Name	endpoint	Rule	Time Created	Action
nginx-ingress	[redacted]	testv2.[redacted].cn-hangzhou.alicontaine... -> http-svc1 foo.bar.com/svc2 -> http-svc2 foo.bar.com/svc3 -> http-svc3	05/02/2018,18:41:55	Details | Update | Delete

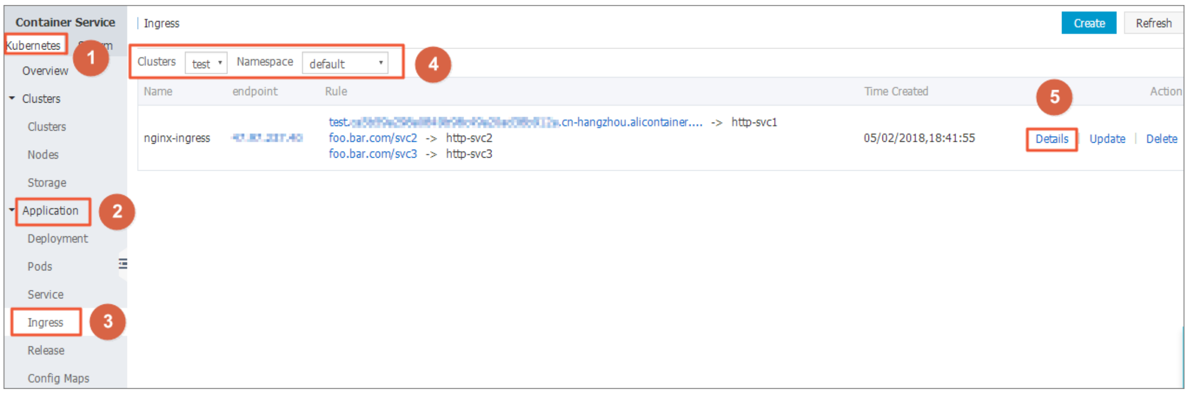
1.7.8 View Ingress details

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 Container Service console*.

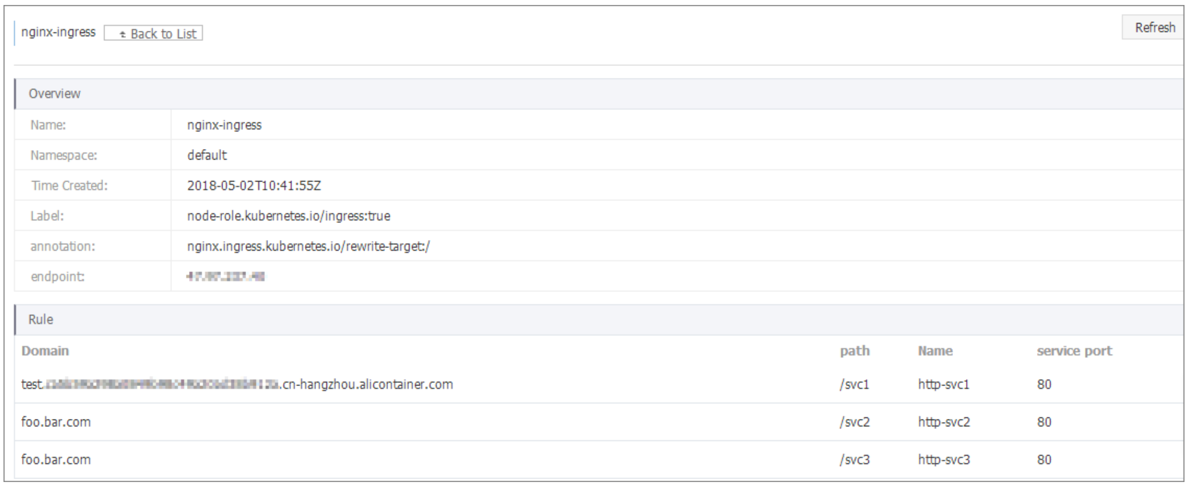
Procedure

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.



Name	endpoint	Rule	Time Created	Action
nginx-ingress	[redacted]	test.[redacted].cn-hangzhou.alicontainer.... -> http-svc1 foo.bar.com/svc2 -> http-svc2 foo.bar.com/svc3 -> http-svc3	05/02/2018,18:41:55	Details | Update | Delete

On the details page, you can view the overview and rules of the Ingress.



Overview

Name:	nginx-ingress
Namespace:	default
Time Created:	2018-05-02T10:41:55Z
Label:	node-role.kubernetes.io/ingress:true
annotation:	nginx.ingress.kubernetes.io/rewrite-target:/
endpoint:	[redacted]

Rule

Domain	path	Name	service port
test.[redacted].cn-hangzhou.alicontainer.com	/svc1	http-svc1	80
foo.bar.com	/svc2	http-svc2	80
foo.bar.com	/svc3	http-svc3	80

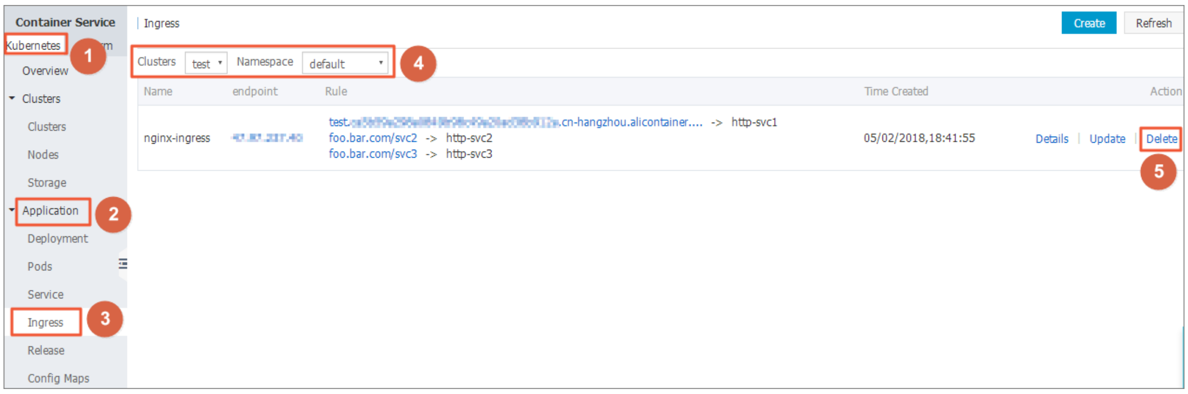
1.7.9 Deleting a route

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 Container Service console*.

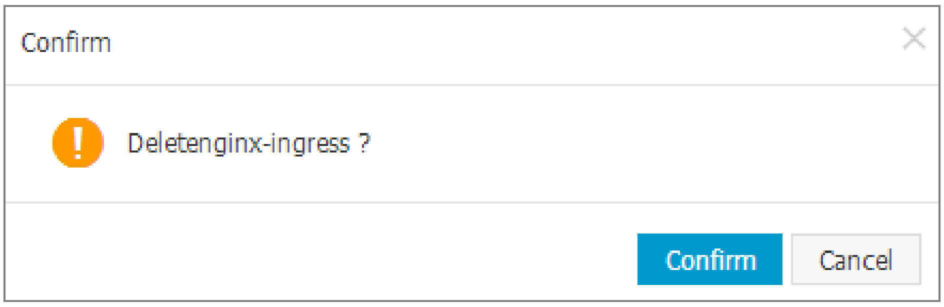
Procedure

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 **Delete** at the right of the Ingress.



Name	endpoint	Rule	Time Created	Action
nginx-ingress	[redacted]	test.[redacted].cn-hangzhou.alicontainer.... -> http-svc1 foo.bar.com/svc2 -> http-svc2 foo.bar.com/svc3 -> http-svc3	05/02/2018,18:41:55	Details | Update | Delete

4. Click **Confirm** in the displayed dialog box.



Confirm

Deletenginx-ingress ?

1.8 Config maps and Secrets

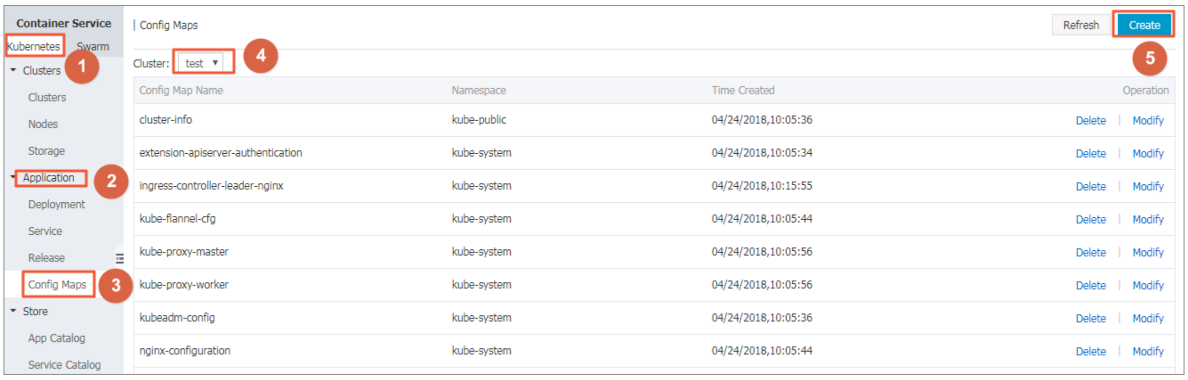
1.8.1 Create a config map

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. Under Kubernetes, click **Application** > **Config Maps** in the left-side navigation pane.

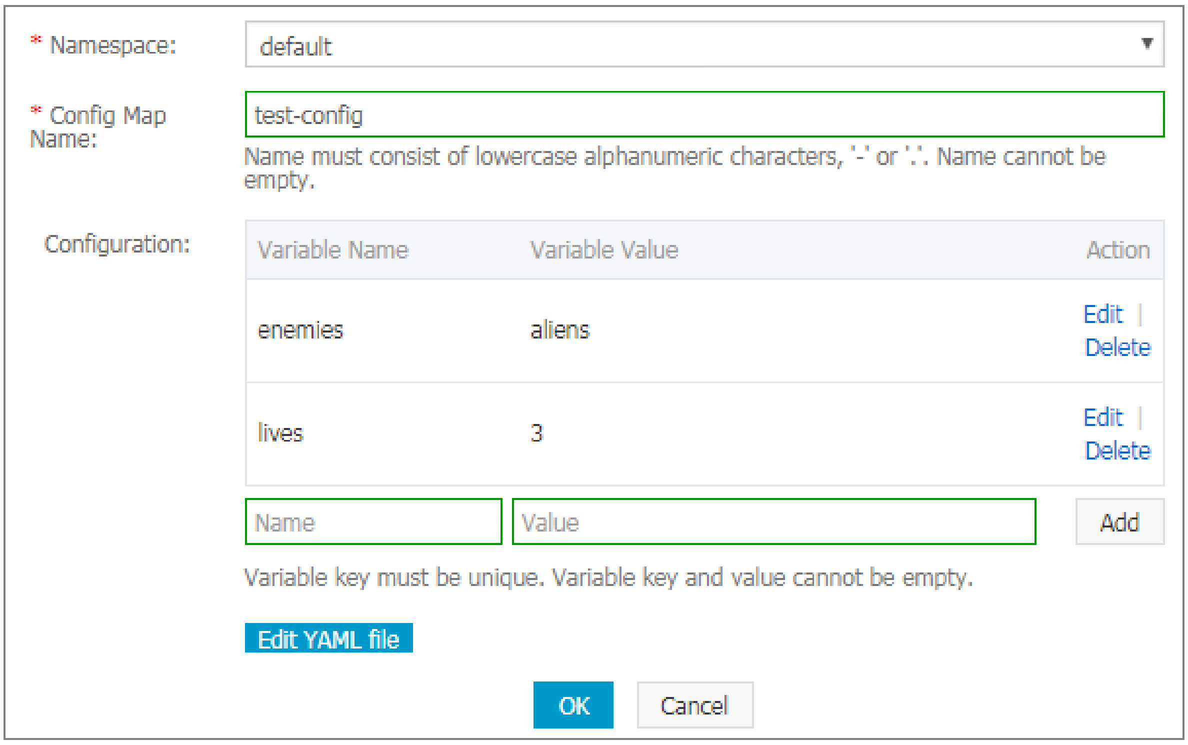
3. Select the cluster from the Cluster drop-down list. Click **Create** in the upper-right corner.



Config Map Name	Namespace	Time Created	Operation
cluster-info	kube-public	04/24/2018,10:05:36	Delete | Modify
extension-apiserver-authentication	kube-system	04/24/2018,10:05:34	Delete | Modify
ingress-controller-leader-nginx	kube-system	04/24/2018,10:15:55	Delete | Modify
kube-flannel-cfg	kube-system	04/24/2018,10:05:44	Delete | Modify
kube-proxy-master	kube-system	04/24/2018,10:05:56	Delete | Modify
kube-proxy-worker	kube-system	04/24/2018,10:05:56	Delete | Modify
kubeadm-config	kube-system	04/24/2018,10:05:36	Delete | Modify
nginx-configuration	kube-system	04/24/2018,10:05:44	Delete | Modify

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**.



* Namespace: default

* Config Map Name: test-config

Name must consist of lowercase alphanumeric characters, '-' or '.'. Name cannot be empty.

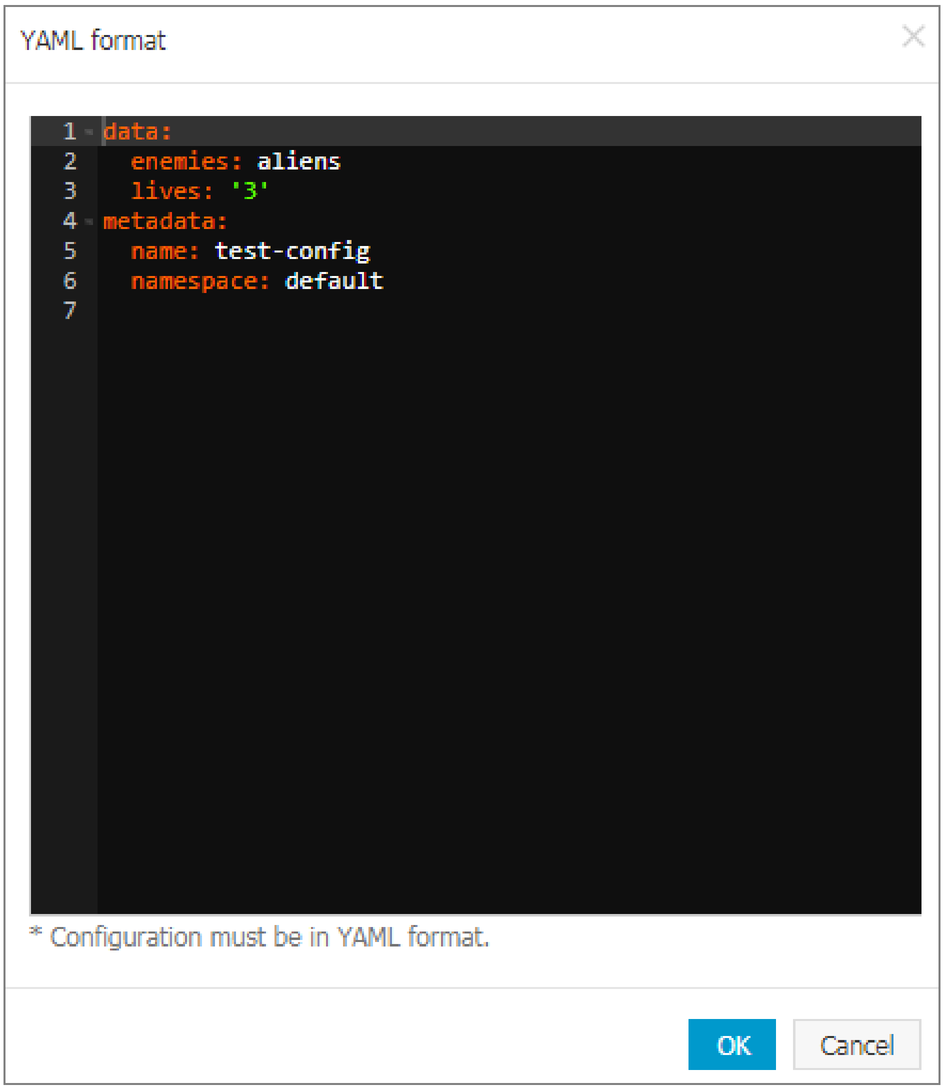
Configuration:

Variable Name	Variable Value	Action
enemies	aliens	Edit | Delete
lives	3	Edit | Delete

Name | Value

Variable key must be unique. Variable key and value cannot be empty.

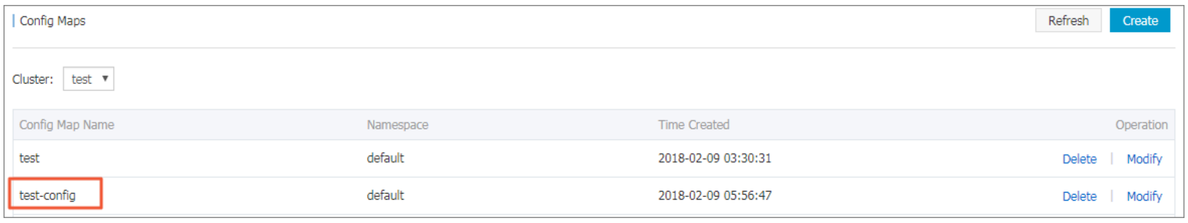
In this example, configure the variables enemies and lives to pass the parameters aliens and 3 respectively.



```
data:
  enemies: aliens
  lives: '3'
metadata:
  name: test-config
  namespace: default
```

* Configuration must be in YAML format.

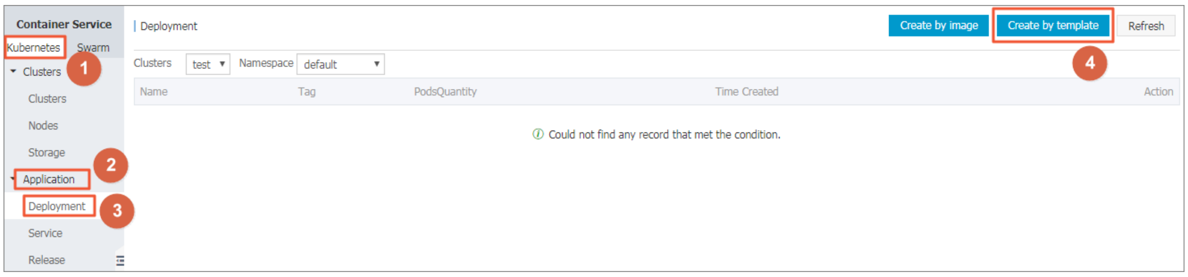
5. You can view the config map test-config on the Config Maps page after clicking **OK**.



Config Map Name	Namespace	Time Created	Operation
test	default	2018-02-09 03:30:31	Delete | Modify
test-config	default	2018-02-09 05:56:47	Delete | Modify

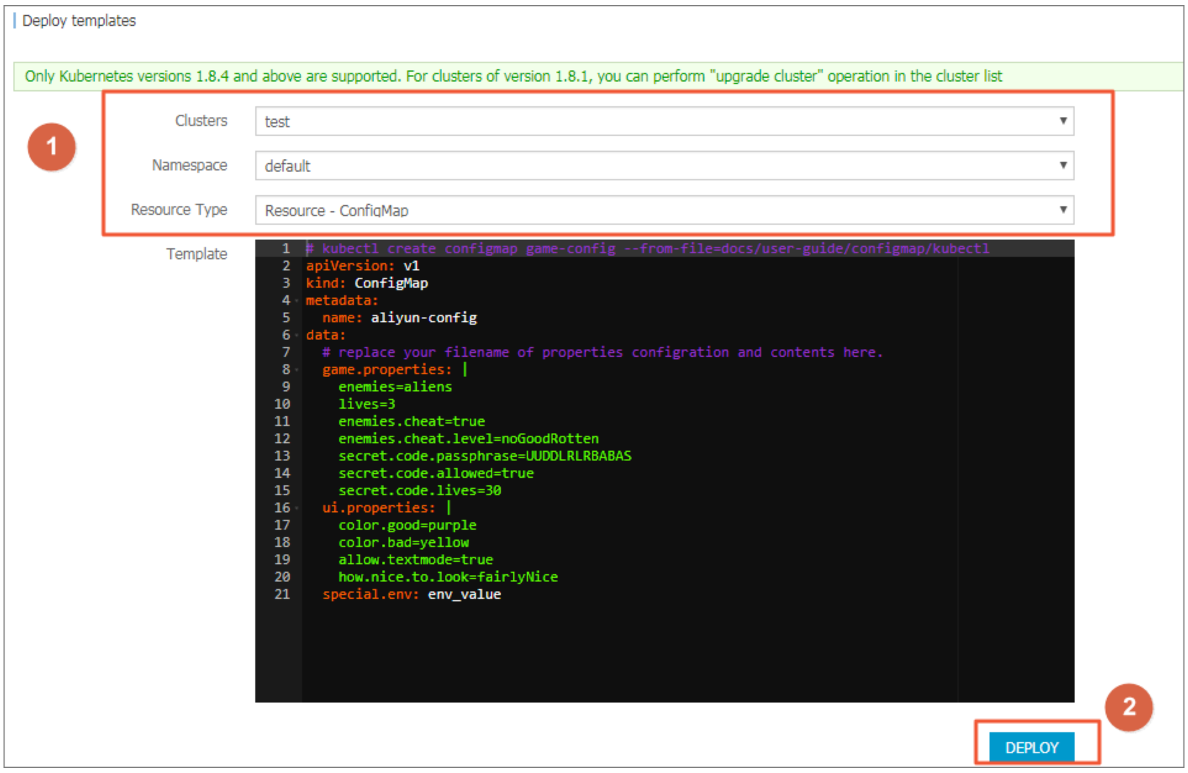
Create a config map by using a template

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



Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

Clusters: test

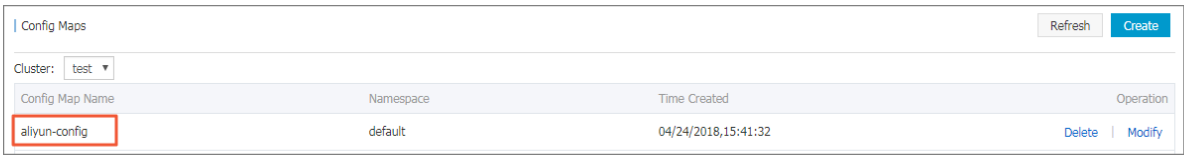
Namespace: default

Resource Type: Resource - ConfigMap

Template

```
# kubectl create configmap game-config --from-file=docs/user-guide/configmap/kubectl
apiVersion: v1
kind: ConfigMap
metadata:
  name: aliyun-config
data:
  # replace your filename of properties configration and contents here.
  game.properties: |
    enemies=aliens
    lives=3
    enemies.cheat=true
    enemies.cheat.level=noGoodRotten
    secret.code.passphrase=UUDDLRLRBABAS
    secret.code.allowed=true
    secret.code.lives=30
  ui.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
    how.nice.to.look=fairlyNice
  special.env: env_value
```

5. After the successful deployment, you can view the config map aliyun-config on the Config Maps page.



Config Map Name	Namespace	Time Created	Operation
aliyun-config	default	04/24/2018,15:41:32	Delete | Modify

1.8.2 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_LEVEL: very` and `SPECIAL_TYPE: 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**.

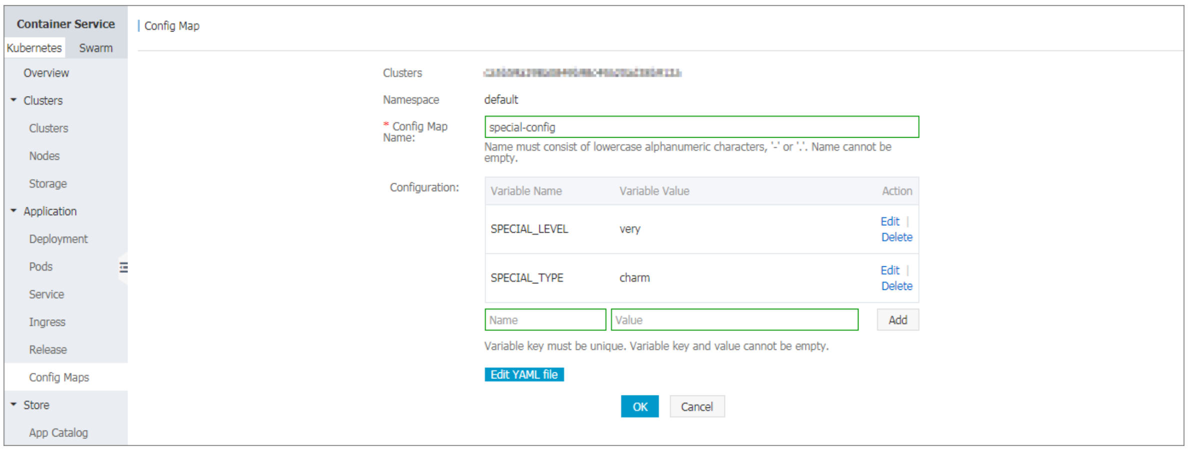
   You can use the following YAML sample template to create a config map.

```
apiVersion: v1
kind: ConfigMap
metadata:
   name: special-config
   namespace: default
data:
   SPECIAL_LEVEL: very
   SPECIAL_TYPE: charm
```

Create a config map on Config Maps page

1. Log on to the *Container Service console*.

2. Under Kubernetes, click**Application** > **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 the Config Map Name. Enter the Variable Name and the Variable Value. Then, click **Add** on the right. Click **OK** after completing the configurations.



Clusters [redacted]

Namespace default

* Config Map Name: special-config

Name must consist of lowercase alphanumeric characters, '-' or '.'. Name cannot be empty.

Configuration:

Variable Name	Variable Value	Action
SPECIAL_LEVEL	very	Edit | Delete
SPECIAL_TYPE	charm	Edit | Delete

Name Value Add

Variable key must be unique. Variable key and value cannot be empty.

Edit YAML file

OK Cancel

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, 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 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_LEVEL_KEY
           valueFrom:                             ##Use valueFrom to
 specify env to reference the value of the config map.
             configMapKeyRef:
               name: special-config              ##The referenced
config map name.
               key: SPECIAL_LEVEL                ##The referenced
config map key.
   restartPolicy: 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.
       - configMapRef:
           name: special-config
   restartPolicy: 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_LEVEL_KEY) $(
SPECIAL_TYPE_KEY)" ]
       env:
         - name: SPECIAL_LEVEL_KEY
           valueFrom:
             configMapKeyRef:
               name: special-config
               key: SPECIAL_LEVEL
         - name: SPECIAL_TYPE_KEY
           valueFrom:
             configMapKeyRef:
               name: special-config
               key: SPECIAL_TYPE
   restartPolicy: Never
```

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.
       volumeMounts:
       - name: config-volume
         mountPath: /etc/config
   volumes:
     - name: config-volume
       configMap:
         name: special-config
   restartPolicy: Never
```

Keys of the config map are output after running the pod.

```
SPECIAL_TYPE
SPECIAL_LEVEL
```

1.8.3 Update a config map

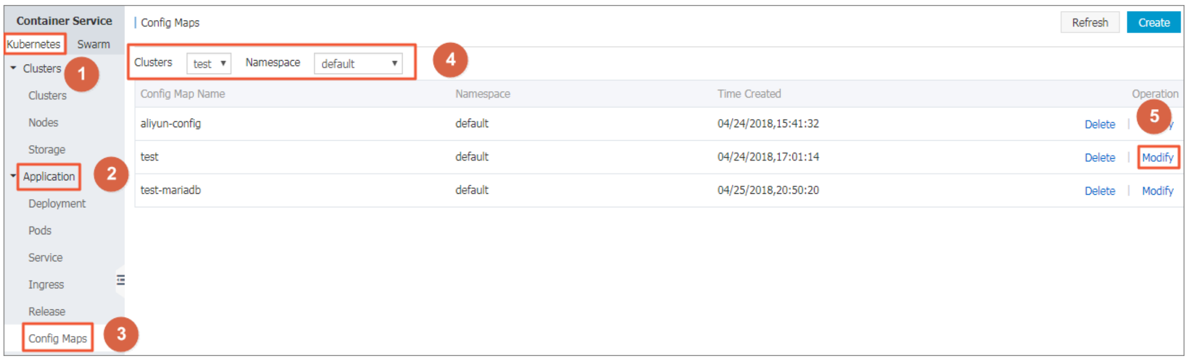
You can modify the configurations of a config map.

Note:

Updating a config map affects applications that use this config map.

Update a config map on Config Maps page

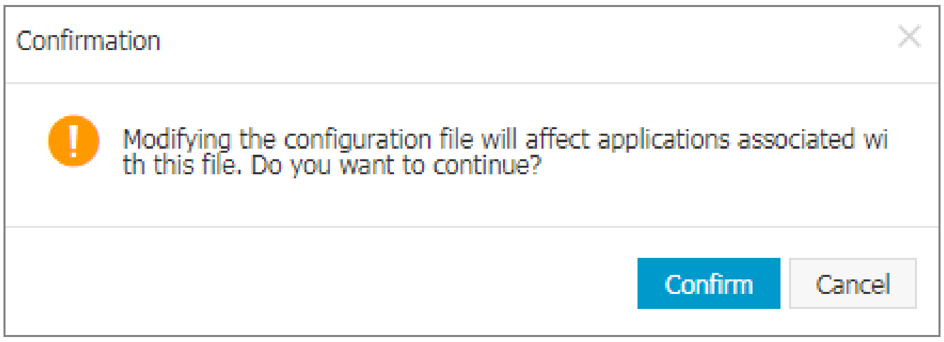
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Config Maps** in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Modify** at the right of the config map.



Clusters test Namespace default

Config Map Name	Namespace	Time Created	Operation
aliyun-config	default	04/24/2018,15:41:32	Delete | Modify
test	default	04/24/2018,17:01:14	Delete | Modify
test-mariadb	default	04/25/2018,20:50:20	Delete | Modify

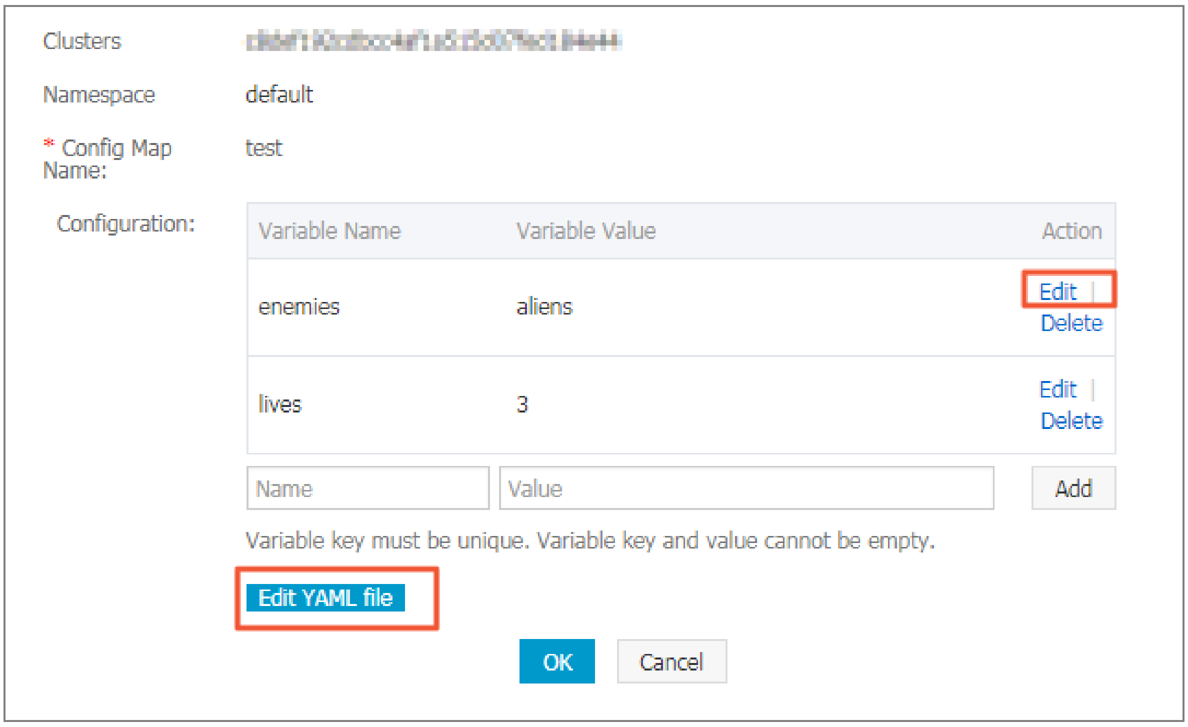
4. Click **Confirm** in the displayed dialog box.



Confirmation

Modifying the configuration file will affect applications associated wi th this file. Do you want to continue?

5. Modify the configurations.
   - Click **Edit** on the right of the configuration you want to modify. Update the configuration and then click **Save**.
   - You can also click **Edit YAML file**. Click **OK** after making the modifications.



Clusters	[illegible]
Namespace	default
* Config Map Name:	test

Configuration:

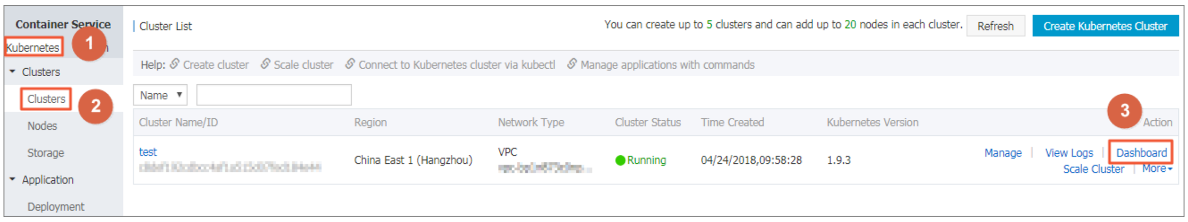
Variable Name	Variable Value	Action
enemies	aliens	Edit | Delete
lives	3	Edit | Delete

Variable key must be unique. Variable key and value cannot be empty.

6. After modifying the configurations, click **OK**.

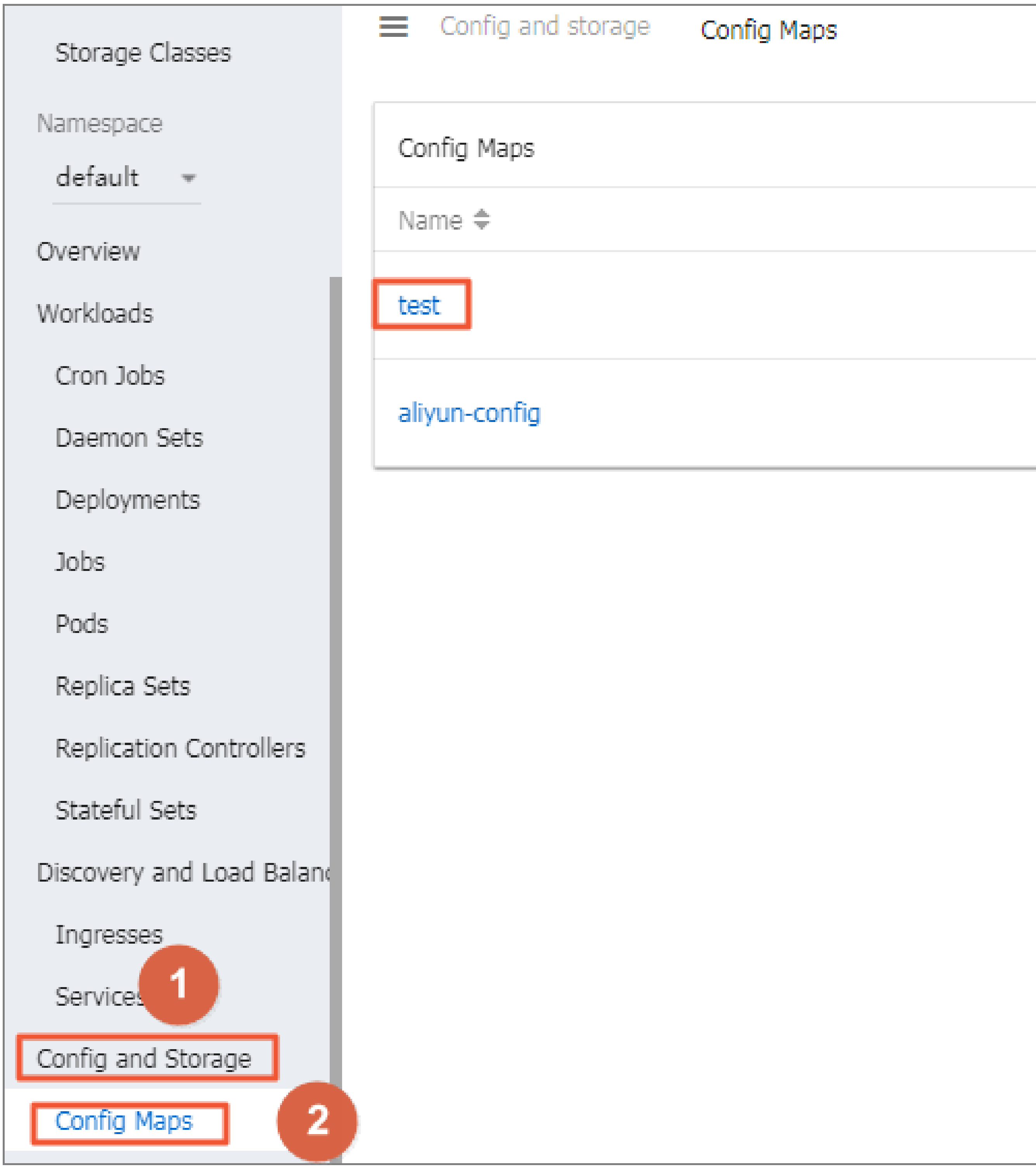
Update a config map in Kubernetes dashboard

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.

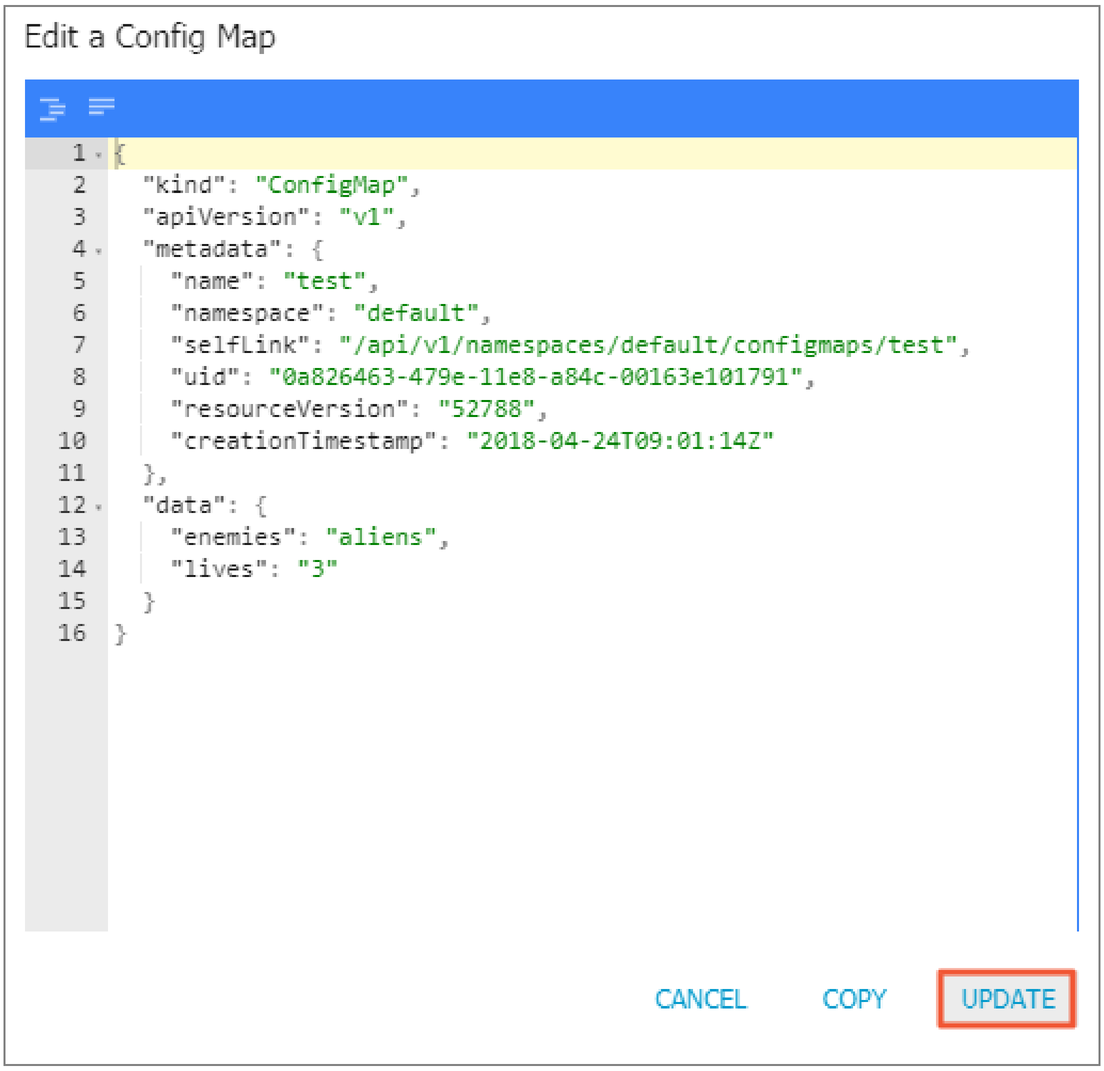

Container Service
Kubernetes
1
Clusters
Clusters
2
Nodes
Storage
Application
Deployment
Cluster List
You can create up to 5 clusters and can add up to 20 nodes in each cluster.
Refresh
Create Kubernetes Cluster
Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands
Name
3
Manage | View Logs | Dashboard
Scale Cluster | More

Cluster Name/ID	Region	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
test	China East 1 (Hangzhou)	VPC	● Running	04/24/2018,09:58:28	1.9.3	Manage | View Logs | Dashboard Scale Cluster | More

4. Under Kubernetes, select a namespace, click **Config and Storage** > **Secrets** in the left-side navigation pane. Select the target secret and click**Actions** > **View/edit YAML**.



5. The Edit a Secret dialog box appears. Modify the configurations and then click **UPDATE**.



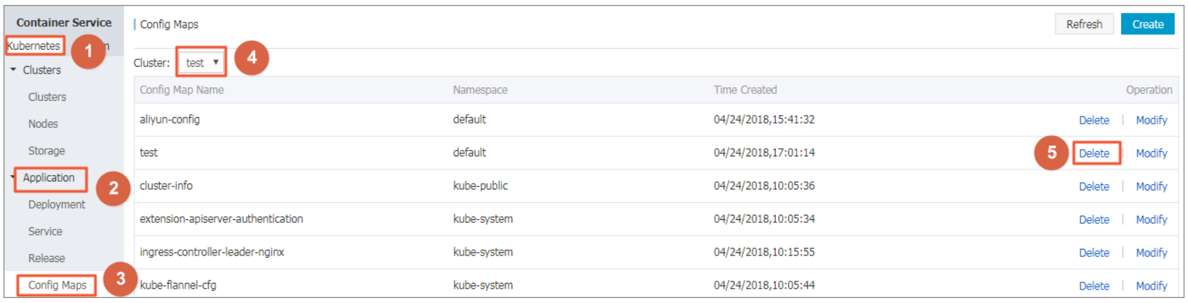
```
{
  "kind": "ConfigMap",
  "apiVersion": "v1",
  "metadata": {
    "name": "test",
    "namespace": "default",
    "selfLink": "/api/v1/namespaces/default/configmaps/test",
    "uid": "0a826463-479e-11e8-a84c-00163e101791",
    "resourceVersion": "52788",
    "creationTimestamp": "2018-04-24T09:01:14Z"
  },
  "data": {
    "enemies": "aliens",
    "lives": "3"
  }
}
```

1.8.4 Delete a config map

You can delete a config map that is no longer in use.

Delete a config map on Config Maps page

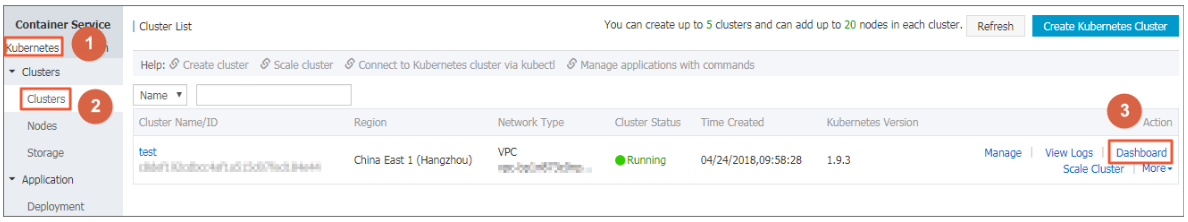
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Config Maps** in the left-side navigation pane.
3. Select the target cluster from the Cluster drop-down list. Click **Delete** at the right of the config map.



Config Map Name	Namespace	Time Created	Operation
aliyun-config	default	04/24/2018,15:41:32	Delete | Modify
test	default	04/24/2018,17:01:14	Delete | Modify
cluster-info	kube-public	04/24/2018,10:05:36	Delete | Modify
extension-apiserver-authentication	kube-system	04/24/2018,10:05:34	Delete | Modify
ingress-controller-leader-nginx	kube-system	04/24/2018,10:15:55	Delete | Modify
kube-flannel-cfg	kube-system	04/24/2018,10:05:44	Delete | Modify

Delete a config map in Kubernetes dashboard

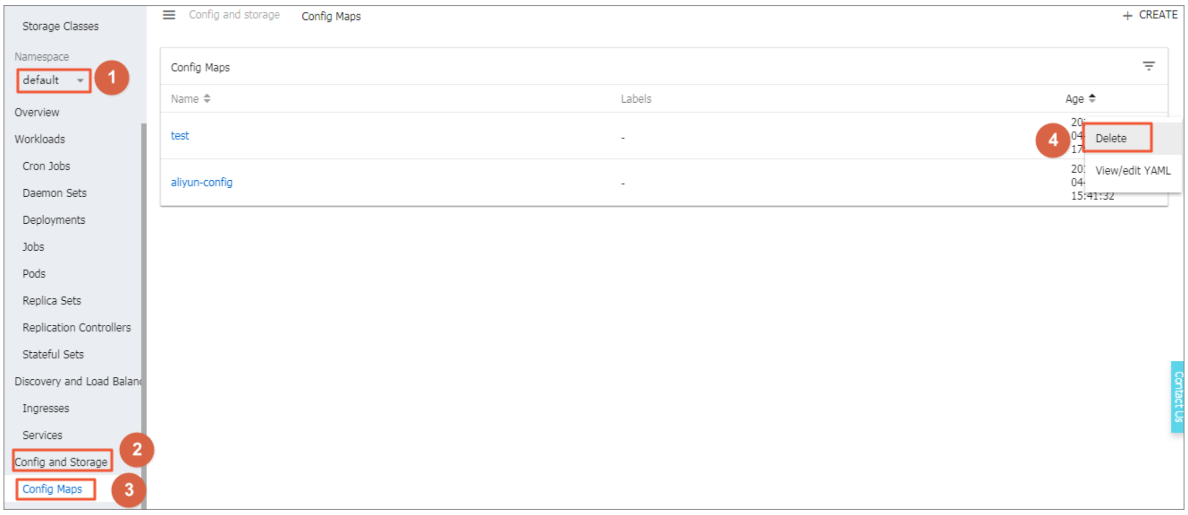
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane.
3. Click **Clusters** in the left-side navigation pane, select the target cluster, and click **Dashboard** on the right.


Container Service
Kubernetes
1
Clusters
Clusters
2
Nodes
Storage
Application
Deployment
Cluster List
Refresh
Create Kubernetes Cluster
Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands
Name
3

You can create up to 5 clusters and can add up to 20 nodes in each cluster.

Cluster Name/ID	Region	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
test [redacted]	China East 1 (Hangzhou)	VPC [redacted]	● Running	04/24/2018,09:58:28	1.9.3	Manage | View Logs | Dashboard Scale Cluster | More

4. Under Kubernetes, select a namespace, click **Config and Storage** > **Secrets** in the left-side navigation pane. Click the actions button on the right and click **Delete** in the drop-down list.



Name	Labels	Age
test	-	20 04 17
aliyun-config	-	20 04 15:41:32

5. Click **Delete** in the displayed dialog box.

1.8.5 Create a secret

Prerequisites

You have created a Kubernetes cluster. 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:

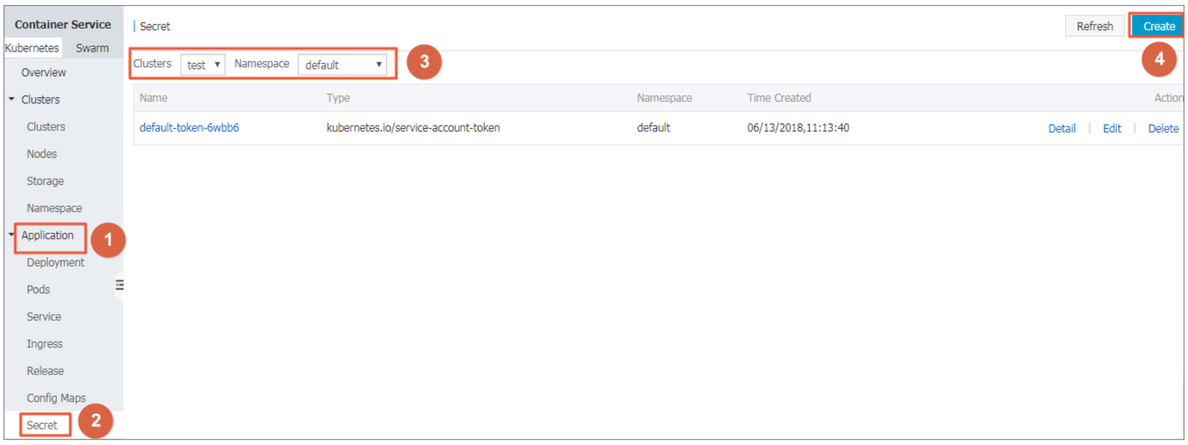
- Service Account: Automatically created by Kubernetes, which is used to access Kubernetes APIs and is automatically mounted to the pod directory *`/run/secrets/kubernetes.io/serviceaccount`*.
- Opaque: Secret in the base64 encoding format, which is used to store sensitive information such as passwords 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 create secrets manually by using command lines. For more information, see *Kubernetes secrets* .

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Secrets** 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.

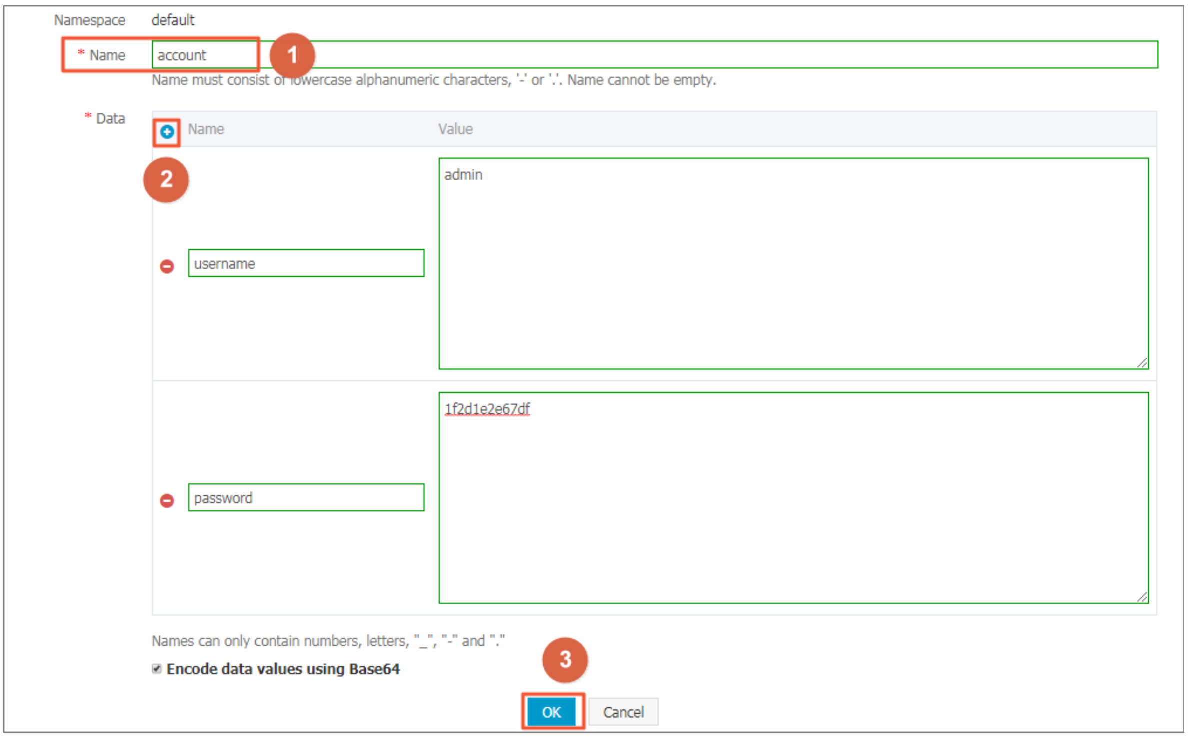


Name	Type	Namespace	Time Created	Action
default-token-6wbb6	kubernetes.io/service-account-token	default	06/13/2018,11:13:40	Detail | Edit | Delete

4. Complete the configurations to create a secret.

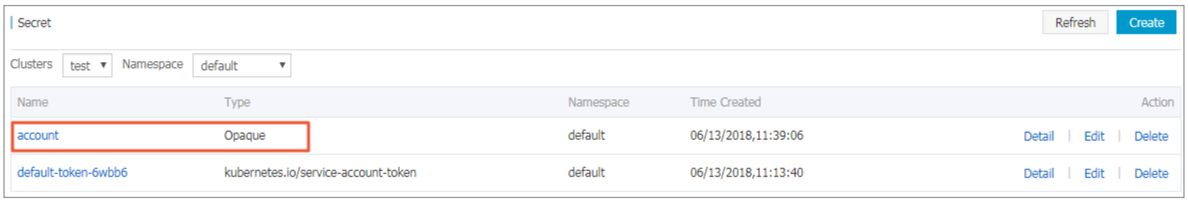
Note:

To enter the clear data of the secret, select the **Encode data values using Base64** check box.


Namespace default
* Name account
1
Name must consist of lowercase alphanumeric characters, '-' or '.'. Name cannot be empty.
* Data
Name
Value
2
username
admin
password
1f2d1e2e67df
Names can only contain numbers, letters, "_", "-" and "."
Encode data values using Base64
3
OK
Cancel

1. Name: Enter the secret name, which must be 1–253 characters long, and can only contain lowercase letters, numbers, hyphens (-), and dots (.).
2. 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` : `1f2d1e2e67df`.
3. Click **OK**.

5. The Secret page appears. You can view the created secret in the secret list.



Name	Type	Namespace	Time Created	Action
account	Opaque	default	06/13/2018,11:39:06	Detail | Edit | Delete
default-token-6wbb6	kubernetes.io/service-account-token	default	06/13/2018,11:13:40	Detail | Edit | Delete

1.8.6 View secret details

You can view the details of a created secret in the Container Service console.

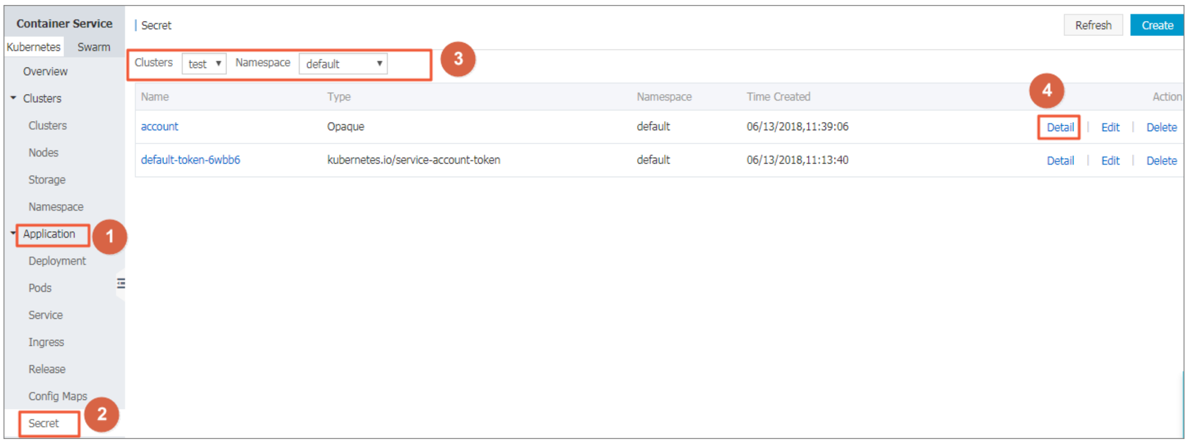
Prerequisites

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

- You have created a secret. For more information, see *Create a secret*.

Procedure

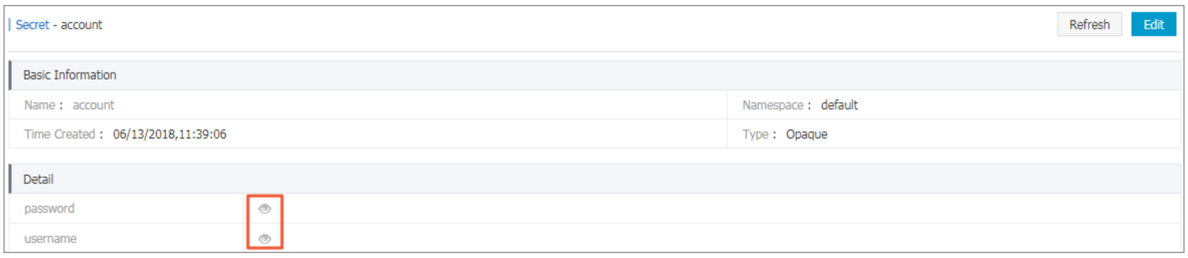
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Secrets** in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Detail** at the right of the secret.



Name	Type	Namespace	Time Created	Action
account	Opaque	default	06/13/2018,11:39:06	Detail | Edit | Delete
default-token-6wbb6	kubernetes.io/service-account-token	default	06/13/2018,11:13:40	Detail | Edit | Delete

4. You can view the basic information of the secret, and the data that the secret contains.

   Click the icon at the right of the data name under Detail to view the clear data.



Name : account
Namespace : default
Time Created : 06/13/2018,11:39:06
Type : Opaque

1.8.7 Update a secret

You can update an existing secret directly in the Container Service console.

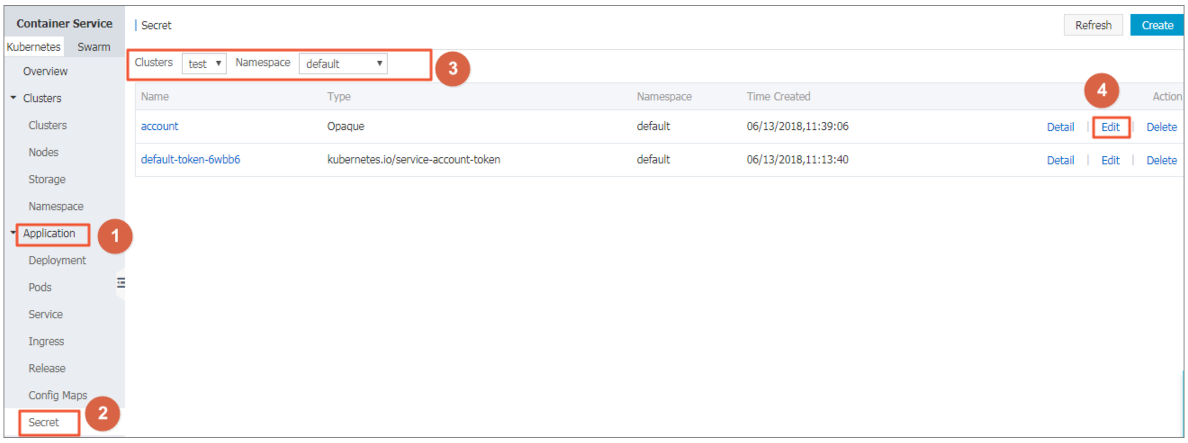
Prerequisites

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

Procedure

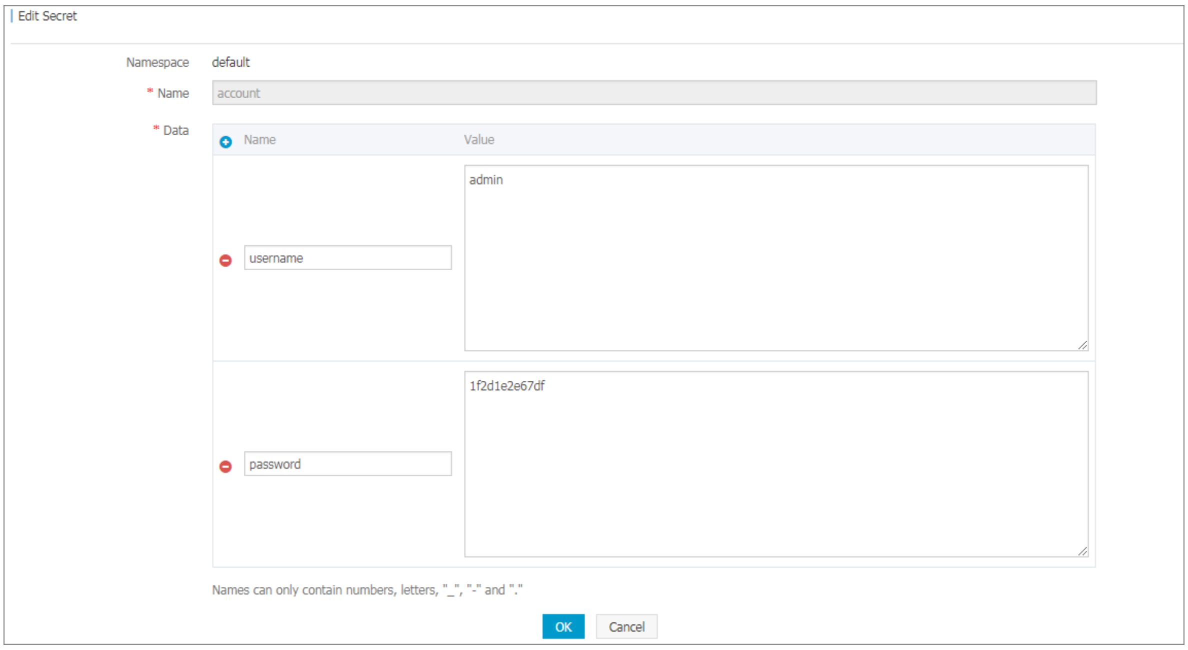
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Secrets** in the left-side navigation pane.

3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Edit** at the right of the secret.



Name	Type	Namespace	Time Created	Action
account	Opaque	default	06/13/2018,11:39:06	Detail | Edit | Delete
default-token-6wbb6	kubernetes.io/service-account-token	default	06/13/2018,11:13:40	Detail | Edit | Delete

4. Update the secret data on the Edit Secret page.



Namespace	default
* Name	account
* Data	

Name	Value
username	admin
password	1f2d1e2e67df

Names can only contain numbers, letters, "_", "-" and "."

5. Click **OK**.

1.8.8 Delete a secret

You can delete an existing secret directly in the Container Service console.

Prerequisites

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

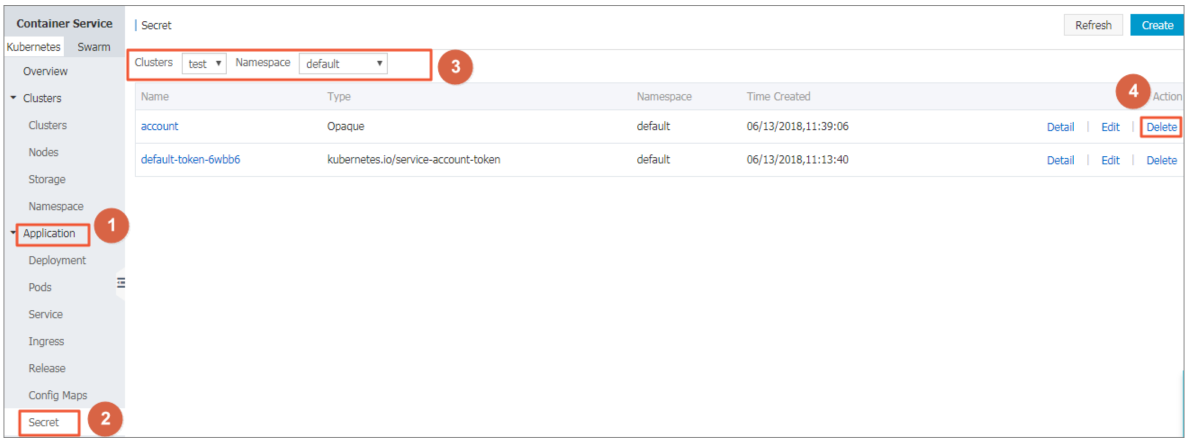
Context

Note:

Do not delete the secret generated when the cluster is created.

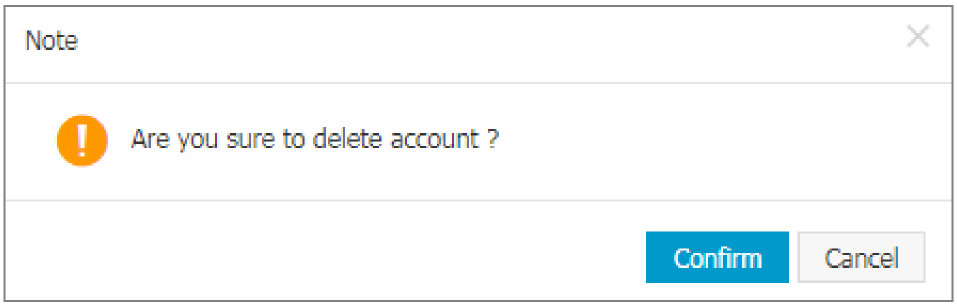
Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Secrets** in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Delete** at the right of the secret.



Name	Type	Namespace	Time Created	Action
account	Opaque	default	06/13/2018,11:39:06	Detail | Edit | Delete
default-token-6wbb6	kubernetes.io/service-account-token	default	06/13/2018,11:13:40	Detail | Edit | Delete

4. Click **Confirm** in the displayed dialog box.



Are you sure to delete account ?

1.9 Storage

1.9.1 Overview

Container Service supports automatically binding Kubernetes pods to Alibaba Cloud cloud disks, NAS, and Object Storage Service (OSS).

Currently, static storage volumes and dynamic storage volumes are supported. See the following table for how each type of data volumes supports the static data volumes and dynamic data volumes.

Alibaba Cloud storage	Static data volume	Dynamic data volume
Alibaba Cloud cloud disk	You can use the cloud disk static storage volumes by: • Using the volume method. • Using PV/PVC.	Supported.
Alibaba Cloud NAS	You can use the NAS static storage volumes by: • Using flexvolume plug-in. — Using the volume method. — Using PV/PVC. • Using NFS drive of Kubernetes.	Supported.
Alibaba Cloud OSS	You can use the OSS static storage volumes by: • Using the volume method. • Using PV/PVC.	Not supported.

1.9.2 Install the plug-in

Deploy the Alibaba Cloud Kubernetes storage plug-in by using the following yaml configurations.

Note:

If your Kubernetes cluster is created before February 6th, 2018, install the Alibaba Cloud Kubernetes storage plug-in before using the data volumes. If your Kubernetes cluster is created after February 6th, 2018, you can directly use the data volumes without installing the Alibaba Cloud Kubernetes storage plug-in.

Limits

Currently, CentOS 7 operating system is supported.

Instructions

- Disable the `--enable-controller-attach-detach` option by using kubelet if you use the flexvolume. By default, Alibaba Cloud Kubernetes clusters have disabled this option.
- Deploy flexvolume in the kube-system user space.

Verify that the installation is complete

On the master node:

- Run the `kubectl get pod -n kube-system | grep flexvolume` command . Output is the list of running pods (number of nodes) .
- Run the `kubectl get pod -n kube-system | grep alicloud-disk-controller` command. Output is the list of running pods.

Installation example

Install flexvolume

```
apiVersion: apps/v1 # for versions before 1.8.0 use extensions/v1beta1
kind: DaemonSet
metadata:
  name: flexvolume
  namespace: kube-system
  labels:
    k8s-volume: flexvolume
spec:
  selector:
    matchLabels:
      name: acs-flexvolume
  template:
    metadata:
      labels:
        name: acs-flexvolume
    spec:
      hostPID: true
      hostNetwork: true
      tolerations:
      - key: node-role.kubernetes.io/master
        operator: Exists
        effect: NoSchedule
      containers:
      - name: acs-flexvolume
        image: registry.cn-hangzhou.aliyuncs.com/acs/flexvolume:v1.9.7
-42e8198
        imagePullPolicy: Always
        securityContext:
          privileged: true
        env:
        - name: ACS_DISK
          value: "true"
        - name: ACS_NAS
          value: "true"
        - name: ACS_OSS
          value: "true"
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: usrdir
          mountPath: /host/usr/
        - name: etcdir
          mountPath: /host/etc/
        - name: logdir
          mountPath: /var/log/alicloud/
      volumes:
      - name: usrdir
        hostPath:
          path: /usr/
      - name: etcdir
        hostPath:
          path: /etc/
      - name: logdir
       hostPath:
          path: /var/log/alicloud/
```

Install Disk provisioner

```
---
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-common
provisioner: alicloud/disk
parameters:
  type: cloud
---
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-efficiency
provisioner: alicloud/disk
parameters:
  type: cloud_efficiency
---
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-ssd
provisioner: alicloud/disk
parameters:
  type: cloud_ssd
---
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-available
provisioner: alicloud/disk
parameters:
  type: available
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: alicloud-disk-controller-runner
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["list", "watch", "create", "update", "patch"]
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: alicloud-disk-controller
  namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: run-alicloud-disk-controller
subjects:
  - kind: ServiceAccount
    name: alicloud-disk-controller
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: alicloud-disk-controller-runner
  apiGroup: rbac.authorization.k8s.io
---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: alicloud-disk-controller
  namespace: kube-system
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: alicloud-disk-controller
    spec:
      tolerations:
      - effect: NoSchedule
        operator: Exists
        key: node-role.kubernetes.io/master
      - effect: NoSchedule
        operator: Exists
        key: node.cloudprovider.kubernetes.io/uninitialized
      nodeSelector:
         node-role.kubernetes.io/master: ""
      serviceAccount: alicloud-disk-controller
      containers:
        - name: alicloud-disk-controller
          image: registry.cn-hangzhou.aliyuncs.com/acs/alicloud-disk-
controller:v1.9.3-ed710ce
          volumeMounts:
            - name: cloud-config
              mountPath: /etc/kubernetes/
            - name: logdir
              mountPath: /var/log/alicloud/
      volumes:
        - name: cloud-config
          hostPath:
            path: /etc/kubernetes/
        - name: logdir
          hostPath:
            path: /var/log/alicloud/
```

1.9.3 Use Alibaba Cloud cloud disks

You can use Alibaba Cloud cloud disk volumes in a Kubernetes cluster of Alibaba Cloud Container Service.

You can mount the cloud disks of Alibaba Cloud to a Kubernetes cluster by using the following two methods:

- *Static volumes*

  You can use a static cloud disk volume in either of the following ways:

  - *Use a static cloud disk volume directly.*
  - *Use a static cloud disk through a PV and PVC.*
- *Dynamic volumes*

Note:

Depending on the type of cloud disk you create, the following requirements must be met:

- The minimum capacity of a basic cloud disk is 5 GiB.
- The minimum capacity of an Ultra disk is 20 GiB.
- The minimum capacity of an SSD cloud disk is 20 GiB.

Static volumes

You can directly use a cloud disk volume of Alibaba Cloud or use the volume through a PV and PVC.

Prerequisites

You must have created a cloud disk in the ECS console. For more information, see *Create a cloud disk*.

Limits

- A cloud disk is a non-shared storage device and can be mounted to only one pod.
- You must have created a cloud disk and obtained the disk ID before using the cloud disk volume. For more information, see *Create a cloud disk*.

- The volumeId indicates the ID of a mounted cloud disk. The volumeName and PV name must be the same as the volumeId.
- In a cluster, a cloud disk can be mounted only to a node that resides in the same zone as the cloud disk.

Use a static cloud disk volume directly

Use the following *disk-deploy.yaml* file to create a pod:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-disk-deploy
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx-flexvolume-disk
        image: nginx
        volumeMounts:
          - name: "d-bp1j17ifxfasvts3tf40"
            mountPath: "/data"
      volumes:
        - name: "d-bp1j17ifxfasvts3tf40"
          flexVolume:
            driver: "alicloud/disk"
            fsType: "ext4"
            options:
              volumeId: "d-bp1j17ifxfasvts3tf40"
```

Use a static cloud disk volume through a PV and PVC

Step 1: Create a cloud disk PV

You can create a cloud disk PV in the Container Service console or by using a YAML file.

Create a PV by using a YAML file

Use the following `disk-pv.yaml` file to create a PV:

Note:

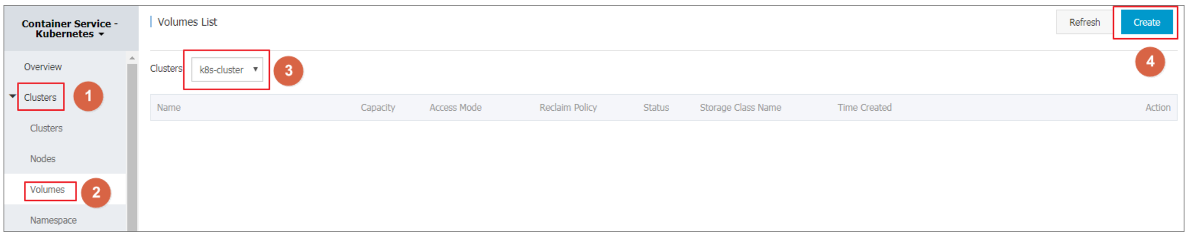
The PV name must be the same as the cloud disk ID.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: d-bp1j17ifxfasvts3tf40
  labels:
    failure-domain.beta.kubernetes.io/zone: cn-hangzhou-b
    failure-domain.beta.kubernetes.io/region: cn-hangzhou
```

```
spec:
  capacity:
    storage: 20Gi
  storageClassName: disk
  accessModes:
    - ReadWriteOnce
  flexVolume:
    driver: "alicloud/disk"
    fsType: "ext4"
    options:
      volumeId: "d-bp1j17ifxfasvts3tf40"
```

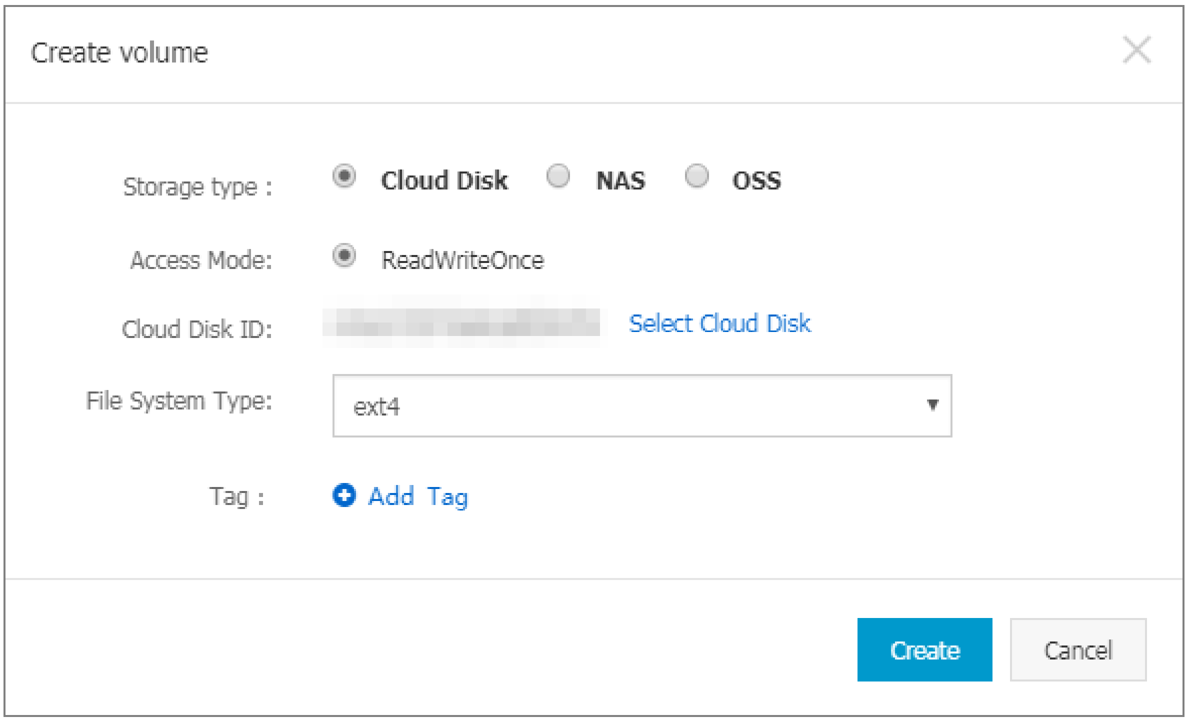
Create a cloud disk volume in the Container Service console

1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose **Clusters** > **Volumes**.
3. Select the target cluster and then click **Create** in the upper-right corner.



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

   - **Storage type**: `Cloud Disk` is used in this example.
   - **Access Mode**: By default, it is set to ReadWriteOnce.
   - **Cloud Disk ID**: We recommend that you select a cloud disk that is in the same region and zone as the cluster.
   - **File System Type**: Select a data type for the data to be stored. The available data types include ext4, ext3, xfs, and vfat. The default setting is ext4.
   - **Tag**: Add tags to the volume.



Create volume

Storage type : [✓ Cloud Disk] NAS OSS

Access Mode: [✓ ReadWriteOnce]

Cloud Disk ID: [redacted] Select Cloud Disk

File System Type: ext4

Tag : Add Tag

5. Click **Create**.

Step 2: Create a PVC

Use the following `disk-pvc.yaml` file to create a PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-disk
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: disk
  resources:
    requests:
      storage: 20Gi
```

Step 3: Create a pod

Use the following *`disk-pod.yaml`* file to create a pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-alicloud-example"
spec:
  containers:
    - name: "nginx"
      image: "nginx"
      volumeMounts:
        - name: pvc-disk
          mountPath: "/data"
  volumes:
```

```
  - name: pvc-disk
    persistentVolumeClaim:
      claimName: pvc-disk
```

Dynamic volumes

To use a dynamic volume, you need to manually create a StorageClass, and specify a cloud disk type through storageClassName in a PVC.

Create a StorageClass

```
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-ssd-hangzhou-b
provisioner: alicloud/disk
parameters:
  type: cloud_ssd
  regionid: cn-hangzhou
  zoneid: cn-hangzhou-b
```

Parameter setting:

- provisioner: Set this parameter to alicloud/disk to indicate that the StorageClass creates a cloud disk by using the provisioner plugin of Alibaba Cloud cloud disks.
- type: Specify the type of a cloud disk by using one the following values: cloud, cloud_efficiency , cloud_ssd, and available. If you set this parameter to available, the system will cycle through cloud_efficiency, cloud_ssd, and cloud in order until one of them takes effect.
- regionid: Set the region in which you want to create a cloud disk.
- reclaimPolicy: Set the policy to reclaim a cloud disk. The default setting is Delete. You can also set this parameter to Retain.
- zoneid: Set the zone in which you want to create a cloud disk.

  Note:
  If you want to create cloud disks in multiple zones, you can set multiple values for the zoneid parameter, for example,

  ```
  zoneid: cn-hangzhou-a,cn-hangzhou-b,cn-hangzhou-c
  ```

- encrypted: (optional) Set whether to encrypt a cloud disk. The default value is `false`. That is, a cloud disk will not be encrypted.

Create a service

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
```

```
  name: disk-ssd
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: alicloud-disk-ssd-hangzhou-b
  resources:
    requests:
      storage: 20Gi
---
kind: Pod
apiVersion: v1
metadata:
  name: disk-pod-ssd
spec:
  containers:
  - name: disk-pod
    image: nginx
    volumeMounts:
      - name: disk-pvc
        mountPath: "/mnt"
  restartPolicy: "Never"
  volumes:
    - name: disk-pvc
      persistentVolumeClaim:
        claimName: disk-ssd
```

Default options

By default, Kubernetes clusters provide the following StorageClasses that can be used in the single-zone clusters:

- alicloud-disk-common, namely, a basic cloud disk.
- alicloud-disk-efficiency, namely, an ultra cloud disk.
- alicloud-disk-ssd, namely, an SSD cloud disk.
- alicloud-disk-available: This StorageClass provides a systematic method of disk selection. Specifically, the system first attempts to create an Ultra cloud disk. If the Ultra cloud disks in the specified zone are sold out, the system tries to create an SSD cloud disk. If the SSD cloud disks are sold out, the system tries to create a basic cloud disk.

Create a multi-instance StatefulSet by using a cloud disk

We recommend that you create a multi-instance StatefulSet through volumeClaimTemplates so that you can dynamically create multiple PVCs and PVs, and connect the PVCs and PVs together.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
  - port: 80
```

```
    name: web
  clusterIP: None
  selector:
    app: nginx
---
apiVersion: apps/v1beta2
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: disk-ssd
          mountPath: /data
  volumeClaimTemplates:
  - metadata:
      name: disk-ssd
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "alicloud-disk-ssd"
      resources:
        requests:
          storage: 20Gi
```

1.9.4 Use NAS file systems of Alibaba Cloud

You can use Alibaba Cloud NAS volumes in a Kubernetes cluster of Container Service.

You can mount a NAS file system of Alibaba Cloud to a Kubernetes cluster as either of the following two types of volumes:

- *Static volumes*

  You can use a static volume in either of the following two ways:

  - Use a static volume through the flexvolume plugin.
    - Use a static volume directly.
    - Use a static volume through a Persistent Volume (PV) and a Persistent Volume Claim (PVC).

    - Use a static volume through the NFS driver of Kubernetes.
- *Dynamic volumes*

Prerequisites

You must have created a NAS file system in the NAS console and added a mount point for a Kubernetes cluster in the file system. You must make sure that the NAS file system and your cluster are in the same VPC.

Static volumes

You can use the Alibaba Cloud NAS file storage service by using the flexvolume plugin provided by Alibaba Cloud or the NFS driver of Kubernetes.

Use a static volume through the flexvolume plugin

With a flexvolume plugin, you can use an Alibaba Cloud NAS volume directly or through a PV and a PVC.

Note:

- NAS: a shared storage system that can provide storage services for multiple pods at the same time.
- server: defines the mount point of a NAS file system.
- path: defines the mount directory that connects to the NAS volume. You can mount a NAS volume to a NAS sub-directory. If no sub-directory exists, the system automatically creates a NAS sub-directory and mounts a NAS volume to the NAS sub-directory.
- vers: defines the version number of the NFS mount protocol. NFS file system versions 3.0 and 4.0 are supported.
- mode:defines the access permission to a mount directory. When the mount directory is the root directory of a NAS file system, the access permission to the root directory cannot be set. If you set the mode parameter for a NAS file system that stores a large amount of data, the process of mounting the NAS file system to a cluster may take an excessive amount of time or even fail.

Use a static volume directly

Use a `nas-deploy.yaml` file to create a pod as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-nas-example"
spec:
  containers:
    - name: "nginx"
      image: "nginx"
      volumeMounts:
        - name: "nas1"
          mountPath: "/data"
  volumes:
    - name: "nas1"
      flexVolume:
        driver: "alicloud/nas"
        options:
          server: "0cd8b4a576-grs79.cn-hangzhou.nas.aliyuncs.com"
          path: "/k8s"
          vers: "4.0"
```

Use a static volume through a PV and a PVC

Step 1: Create a PV

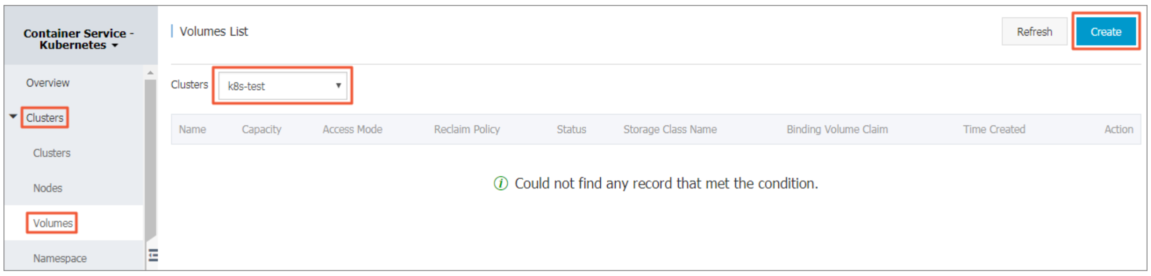
You can create a NAS volume by using a YAML file or create a NAS volume in the Alibaba Cloud Container Service console.

- Create a PV by using a YAML file.

  Use a `nas-pv.yaml` file to create a PV as follows:

  ```
  apiVersion: v1
  kind: PersistentVolume
  metadata:
    name: pv-nas
  spec:
    capacity:
      storage: 5Gi
    storageClassName: nas
    accessModes:
      - ReadWriteMany
    flexVolume:
      driver: "alicloud/nas"
      options:
        server: "0cd8b4a576-uih75.cn-hangzhou.nas.aliyuncs.com"
        path: "/k8s"
        vers: "4.0"
  ```
- Create a NAS volume in the Container Service console.

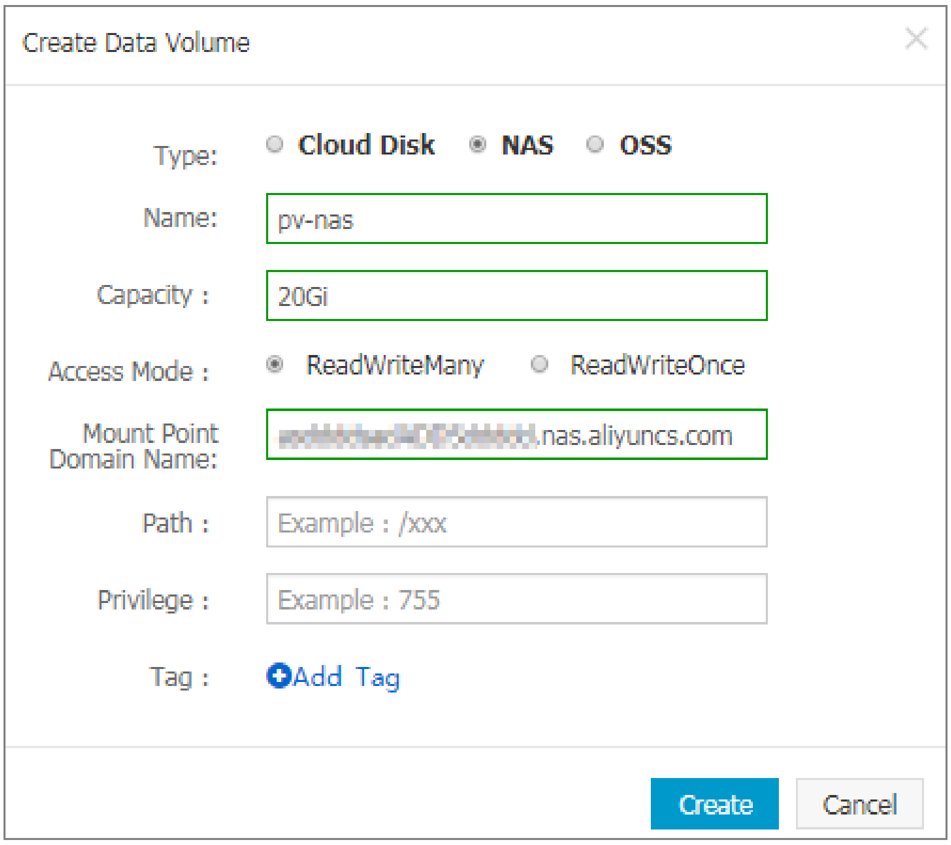
  1. Log on to the *Container Service console*.
  2. In the left-side navigation pane under Kubernetes, choose **Clusters** > **Volumes**.
  3. Select the target cluster from the cluster drop-down list and then click **Create** in the upper-right corner.



Could not find any record that met the condition.

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

   - **Storage type**: NAS is selected in this example.
   - **Name**: Customize a volume name. The volume name must be unique in the cluster. In this example, pv-nas is set as the volume name.
   - **Capacity**: Set the volume capacity. Make sure that the volume capacity does not exceed the NAS file system capacity.
   - **Access Mode**: By default, it is set to ReadWriteOnce.
   - **Mount Point Domain Name**: Enter the mount address of the mount point that is used to mount the NAS file system to the Kubernetes cluster.
   - **Path**: sub-directory under the NAS path, which starts with a forward slash (/). If you specify a sub-directory, your volume will be mounted to the sub-directory.
     - If no sub-directory exists in the root directory of a NAS file system, the system automatically creates a sub-directory by default.
     - This parameter is optional. A NAS volume is mounted to the root directory of a NAS file system by default.
   - **Privilege**: Set the access permission to the mount directory. For example, you can set this parameter to 755, 644, or 777.
     - You can set this parameter only if you mount a NAS volume to the NAS sub-directory . This parameter cannot be set if you mount a NAS volume to the NAS root directory.
     - This parameter is optional. By default, the original access permission to a NAS file system is used.
   - **Tag**: Add tags to the volume.



Create Data Volume

Type:	Cloud Disk [✓ NAS] OSS
Name:	pv-nas
Capacity :	20Gi
Access Mode :	[✓ ReadWriteMany] ReadWriteOnce
Mount Point Domain Name:	[redacted].nas.aliyuncs.com
Path :	Example : /xxx
Privilege :	Example : 755
Tag :	Add Tag

5. Click **Create**.

Step 2: Create a PVC

Use a *nas-pvc.yaml* file to create a PVC as follows:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-nas
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: nas
  resources:
    requests:
      storage: 5Gi
```

Step 3: Create a pod

Use a *nas-pod.yaml* file to create a pod as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-nas-example"
spec:
  containers:
    - name: "nginx"
```

```
      image: "nginx"
      volumeMounts:
          - name: pvc-nas
            mountPath: "/data"
  volumes:
  - name: pvc-nas
    persistentVolumeClaim:
        claimName: pvc-nas
```

Use the Kubernetes NFS driver

Note:

Alibaba Cloud NAS supports NFS 3.0 and NFS 4.0. You must specify a valid NFS version when you create a NAS volume.

Step 1: Create a NAS file system

Log on to the *NAS console* to create a NAS file system.

Note:

You must ensure that the NAS file system and your cluster are in the same region.

For example, assume that the mount point of your NAS file system is `055f84ad83-ixxxx.cn-hangzhou.nas.aliyuncs.com`.

Step 2: Create a PV

You can create a NAS volume by using an orchestration template or the Alibaba Cloud Container Service console.

- **Use an orchestration template to create a NAS volume**

  Use a *`nas-pv.yaml`* file to create a PV.

  Run the following command to create a NAS PV:

```
root@master # cat << EOF |kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nas
spec:
  capacity:
    storage: 8Gi
  accessModes:
    - ReadWriteMany
  mountOptions:
    - noresvport
    - nfsvers=4.0
  persistentVolumeReclaimPolicy: Retain
  nfs:
    path: /
    server: 055f84ad83-ixxxx.cn-hangzhou.nas.aliyuncs.com
EOF
```

- **Create a NAS volume in the Container Service console**

  For more information, see *Use a PV and a PVC*.

Step 2: Create a PVC

Create a PVC to request to bind the PV.

```
root@master # cat << EOF | kubectl apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
     name: nasclaim
spec:
     accessModes:
       - ReadWriteMany
     resources:
       requests:
         storage: 8Gi
EOF
```

Step 3: Create a pod

Create an application to declare to mount and use the volume.

```
root@master # cat << EOF |kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
     name: mypod
spec:
     containers:
       - name: myfrontend
         image: registry.aliyuncs.com/spacexnice/netdia:latest
         volumeMounts:
         - mountPath: "/var/www/html"
           name: mypd
     volumes:
       - name: mypd
         persistentVolumeClaim:
           claimName: nasclaim
EOF
```

The NAS file system is successfully mounted to the application that runs on the pod.

Dynamic volumes

To use a dynamic NAS volume, you need to manually install a driver plugin and configure a NAS mount point.

Note:

To dynamically generate a NAS volume is to automatically generate a directory in an existing NAS file system. This directory is defined as the target volume.

Install a plugin

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: alicloud-nas
mountOptions:
- vers=4.0
provisioner: alicloud/nas
reclaimPolicy: Retain

---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: alicloud-nas-controller
  namespace: kube-system
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: alicloud-nas-controller
    spec:
      tolerations:
      - effect: NoSchedule
        operator: Exists
        key: node-role.kubernetes.io/master
      - effect: NoSchedule
        operator: Exists
        key: node.cloudprovider.kubernetes.io/uninitialized
      nodeSelector:
        node-role.kubernetes.io/master: ""
      serviceAccount: admin
      containers:
        - name: alicloud-nas-controller
          image: registry.cn-hangzhou.aliyuncs.com/acs/alicloud-nas-
controller:v3.1.0-k8s1.11
          volumeMounts:
          - mountPath: /persistentvolumes
            name: nfs-client-root
          env:
            - name: PROVISIONER_NAME
              value: alicloud/nas
            - name: NFS_SERVER
              value: 0cd8b4a576-mmi32.cn-hangzhou.nas.aliyuncs.com
            - name: NFS_PATH
              value: /
      volumes:
      - name: nfs-client-root
        flexVolume:
          driver: alicloud/nas
          options:
            path: /
            server: 0cd8b4a576-mmi32.cn-hangzhou.nas.aliyuncs.com
            vers: "4.0"
```

Use the dynamic volume

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  volumeClaimTemplates:
  - metadata:
      name: html
    spec:
      accessModes:
        - ReadWriteOnce
      storageClassName: alicloud-nas
      resources:
        requests:
          storage: 2Gi
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:alpine
        volumeMounts:
        - mountPath: "/usr/share/nginx/html/"
          name: html
```

1.9.5 Use Alibaba Cloud OSS volumes

You can use Alibaba Cloud OSS volumes in a Kubernetes cluster of Alibaba Cloud Container Service.

Specifically, you can only use static OSS volumes. Dynamic OSS volumes are not supported. You can use a static OSS volume in either of the following two ways:

- Use an OSS bucket through a volume.
- Use an OSS bucket through a Persistent Volume (PV) and a Persistent Volume Claim (PVC).

Prerequisites

You must have created a bucket in the OSS console.

OSS parameter setting

- OSS: OSS is a shared storage system that can provide storage services to multiple pods at the same time.

- bucket: Only buckets can be mounted to a Kubernetes cluster. The sub-directories or files under a bucket cannot be mounted to a Kubernetes cluster.
- url: Specify an OSS endpoint, namely, the domain name used to mount an OSS bucket to a cluster.
- akId: Enter your Access Key ID.
- akSecret: Enter your Access Key Secret.
- otherOpts: Customize other parameters in the format of `-o *** -o ***`.

Notices

- If your Kubernetes cluster is created before February 6th, 2018, *Install the plug-in* before using a volume. Before you can use the OSS volume, you must first create a secret and then enter your Access Key information into the secret when you deploy the flexvolume service.
- If you upgrade a Kubernetes cluster of Container Service or restart a kubelet, the Kubernetes cluster network is reset and the mounted OSS volumes will be remounted to the cluster. In this case, you need to recreate the pod that use the OSS volumes. To solve this problem more efficiently, you can configure a health check in the YAML file of the pod so that the pod can automatically restart when the OSS volumes are remounted.

Use a static OSS volume

Use an OSS bucket through a volume

Use a *`oss-deploy.yaml`* file to create a pod.

Note:

If you upgrade a Kubernetes cluster of Container Service or restart a kubelet, the Kubernetes cluster network is reset. To guarantee that the system automatically restarts the container when the OSS directory within the container becomes unavailable, configure the `livenessProbe` health check for the container.

The parameter description of `livenessProbe` is as follows:

- `command`, health check command. The format is,

```
command:
- h
-c
```

```
- cd /data
```

where, `- cd /data` is the OSS directory within the container. You only need to one directory even if multiple directories exist within the container.

- `initialDelaySeconds` indicates the number of seconds for which the first probe must wait after the container is started.
- `periodSeconds` indicates the interval at which probes are performed.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-oss-deploy
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx-flexvolume-oss
        image: nginx
        volumeMounts:
          - name: "oss1"
            mountPath: "/data"
        livenessProbe:
          exec:
            command:
            - sh
            - -c
            - cd /data
          initialDelaySeconds: 30
          periodSeconds: 30
      volumes:
        - name: "oss1"
          flexVolume:
            driver: "alicloud/oss"
            options:
              bucket: "docker"
              url: "oss-cn-hangzhou.aliyuncs.com"
              akId: ***
              akSecret: ***
              otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

Use a PV and a PVC

Step 1: Create a PV

You can create a PV by using a YAML file or the Container Service console.

Use a YAML file to create a PV

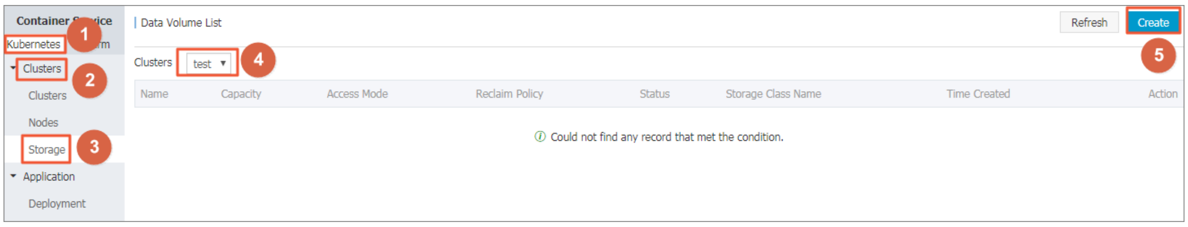
Use a *oss-pv.yaml* file to create a PV as follows:

```
apiVersion: v1
```

```
kind: PersistentVolume
metadata:
  name: pv-oss
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  storageClassName: oss
  flexVolume:
    driver: "alicloud/oss"
    options:
      bucket: "docker"
      url: "oss-cn-hangzhou.aliyuncs.com"
      akId: ***
      akSecret: ***
      otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

Create an OSS volume in the Container Service console

1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose **Clusters** > **Volumes**.
3. Select the target cluster from the cluster drop-down list and then click **Create** in the upper-right corner.



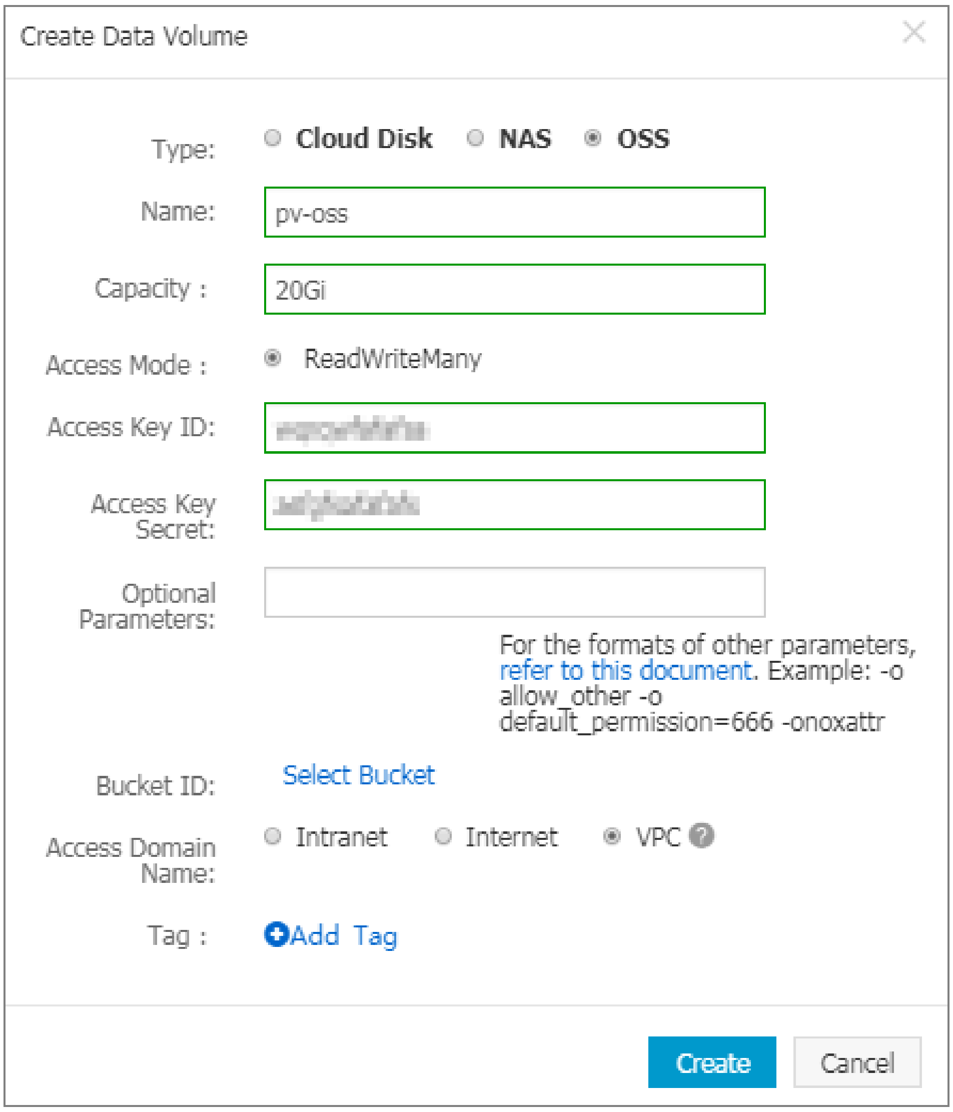
Name	Capacity	Access Mode	Reclaim Policy	Status	Storage Class Name	Time Created	Action

Could not find any record that met the condition.

4. In the displayed dialog box, set the volume parameters.
   - **Storage type**: OSS is selected in this example.
   - **Name**: Customize a volume name. The volume name must be unique in the cluster. In this example, pv-oss is set as the volume name.
   - **Capacity**: Set the volume capacity.
   - **Access Mode**: By default, it is set to ReadWriteMany.
   - **AccessKey ID** and **AccessKey Secret**: Use these two parameters to specify the Access Key used to access OSS.
   - **Bucket ID**: Select an OSS bucket name. Click **Select Bucket**. In the displayed dialog box, select the target bucket and click**Select**.
   - **Access Domain Name**. If the selected bucket and the cluster ECS instances are in different regions, you need to select**Internet**. If they are in the same region, your choice is dependent

on your cluster network type. If your cluster uses a VPC, you need to select **VPC**; if your cluster uses a classic network, you need to select **Intranet**.

- **Tag**: Add tags to the volume.


Create Data Volume

Type:	Cloud Disk NAS [✓ OSS]
Name:	pv-oss
Capacity :	20Gi
Access Mode :	[✓ ReadWriteMany]
Access Key ID:	[redacted]
Access Key Secret:	[redacted]
Optional Parameters:	For the formats of other parameters, refer to this document. Example: -o allow_other -o default_permission=666 -onoxattr
Bucket ID:	Select Bucket
Access Domain Name:	Intranet Internet [✓ VPC]
Tag :	Add Tag

5. Click **Create**.

Step 2: Create a PVC

Use a *`oss-pvc.yaml`* file to create a PVC as follows:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-oss
spec:
  storageClassName: oss
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```

Step 3: Create a pod

Use a *oss-pod.yaml* file to create a pod.

Note:

If you upgrade a Kubernetes cluster of Container Service or restart a kubelet, the Kubernetes cluster network is reset. To guarantee that the system automatically restarts the container when the OSS directory within the container becomes unavailable, configure the `livenessProbe` health check for the container.

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-oss-example"
spec:
  containers:
    - name: "nginx"
      image: "nginx"
      volumeMounts:
          - name: pvc-oss
            mountPath: "/data"
      livenessProbe:
        exec:
          command:
          - sh
          - -c
          - cd /data
        initialDelaySeconds: 30
        periodSeconds: 30
  volumes:
  - name: pvc-oss
    persistentVolumeClaim:
        claimName: pvc-oss
```

Use a dynamic OSS volume

Dynamic OSS volumes are not supported.

1.9.6 Create a persistent storage volume claim

You can create a persistent storage volume claim (PVC) by using the Container Service console.

Prerequisites

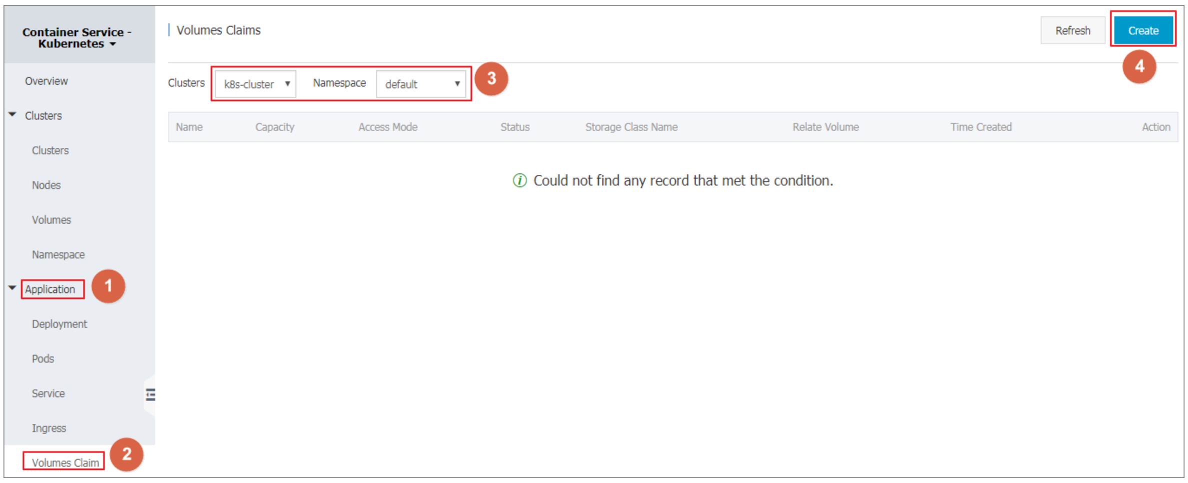
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a storage volume. In this example, use a cloud disk to create a cloud storage volume. For more information, see *Use Alibaba Cloud cloud disks*.

By default, the storage claim is bound to the storage volume depending on the label `alicloud-pvname`. When the data volume is created by using the Container Service console, the storage volume is labeled by default. If the storage volume label does not exist, you must add a label before you select to bound this storage volume.

Context

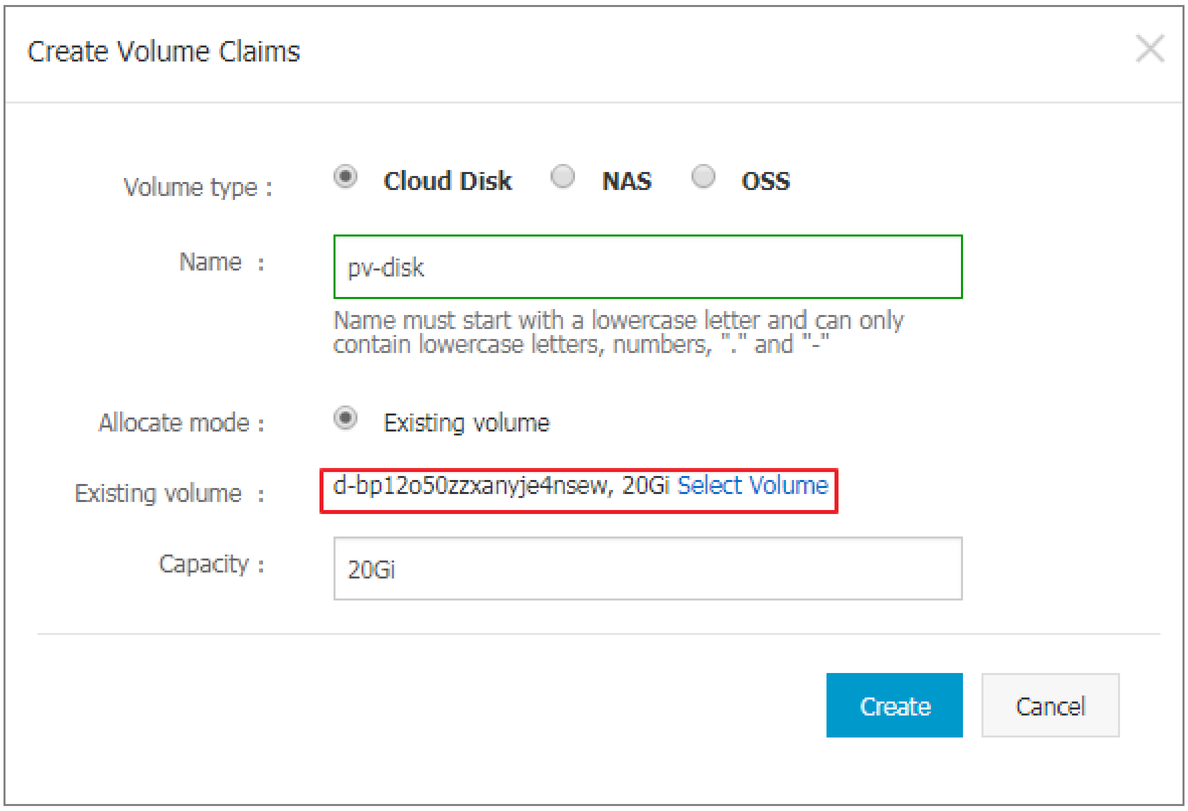
Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Volumes Claim** in the left-side navigation pane to enter the Volumes Claims list page.
3. Select the target cluster and namespace, and click **Create** in the upper-right corner.



Could not find any record that met the condition.

4. Complete the configurations in the Create Volume Claim dialog box, and click **Create**.


Create Volume Claims

Volume type : [✓ Cloud Disk] NAS OSS

Name : pv-disk

Name must start with a lowercase letter and can only contain lowercase letters, numbers, "." and "-"

Allocate mode : [✓ Existing volume]

Existing volume : d-bp12o50zzxanyje4nsew, 20Gi Select Volume

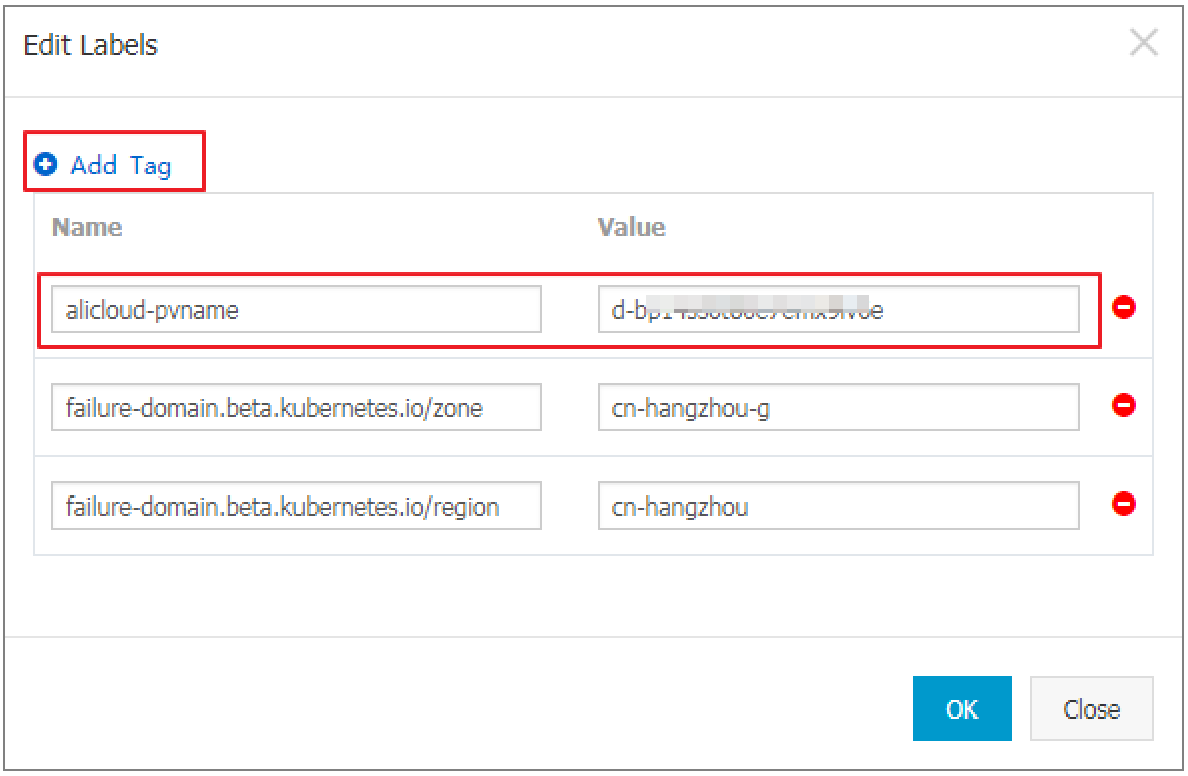
Capacity : 20Gi

- **Volume claim type**: Consistent with storage volume, including cloud disk, NAS, and OSS types.
- **Name**: Enter the storage volume claim name.
- **Distribution mode**: Currently, only existing storage volumes are supported.
- **Existing storage volume**: Select to bind the storage volume of this type.
- **Total**: Claim usage, cannot be greater than the total amount of storage volumes.

Note:

If a storage volume already exists in your cluster and is not used, but cannot be found in **Select Existing Storage Volume**, maybe the `alicloud-pvname` label is not defined.

If you cannot find an available storage volume, you can click **Clusters** > **Volumes** in the left-side navigation pane. Find the target storage volume, click **Label Management** on the right. Add the corresponding label `alicloud-pvname`, the value is the name of the storage volume. The cloud storage volume defaults to the cloud disk ID as the name of the storage volume.


Edit Labels

Add Tag

Name	Value
alicloud-pvname	d-bp[redacted]e
failure-domain.beta.kubernetes.io/zone	cn-hangzhou-g
failure-domain.beta.kubernetes.io/region	cn-hangzhou

5. Return to the Volumes Claims list, you can see that the newly created storage claim appears in the list.

1.9.7 Using persistent storage volume claim

On the Container Service console, use an image or a template to deploy an application, so that you can use a persistent storage volume claim. In this example, an image is used to create an application. If you want to use a persistent storage volume claim with the template, see *#unique_124*.

Prerequisites

- You have created a Kubernetes cluster. For more information, see *#unique_31*.
- If you have already created a storage volume claim, use the cloud disk to create a cloud disk storage volume claim PVC disk. For more information, see *#unique_125*.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane. Enter the Deployment List page and click **Create by image** in the upper-right corner.

3. On the Basic Information page, configure the application name, deploy the cluster, and the namespace. Then click **Next**.

4. On the Application Configuration page, select Image. Then configure the cloud storage type of data volume, cloud disk, NAS, and OSS types are supported. In this example, use the cloud storage volume claim and click **Next**.

5. See *#unique_81* to configure the test-nginx application, and click **Create**.
6. After the application is created, click **Apply** > **Container Group** in the left-side navigation pane. Find the container group to which the application belongs, and click **Details**.

7. On the Container Group details page, click **Storage** to view the container group is properly bound to the PVC disk.

1.10 Logs

1.10.1 Overview

The Alibaba Cloud Container Service Kubernetes cluster provides you with multiple methods to manage application logs.

- With the open-source Fluentd-pilot project provided by Alibaba Cloud Container Service, you can conveniently *#unique_128* to better use the features provided by Alibaba Cloud Log Service such as log statistics and analysis.
- *#unique_129*.
- *#unique_130*.

1.10.2 View cluster logs

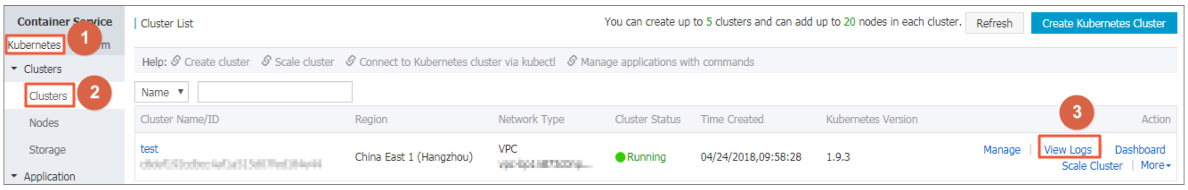
Context

You can view the cluster operation logs by using the simple log service of Container Service.

Procedure

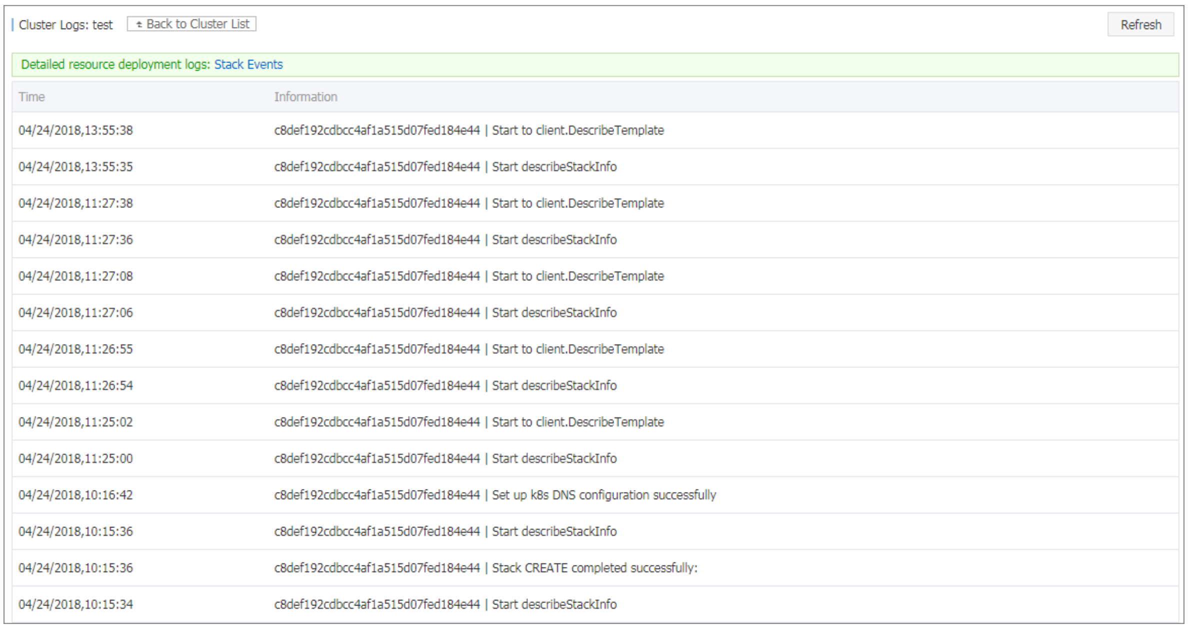
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane.

3. Click **View Logs** at the right of the cluster.


Container Service
Kubernetes
1
Clusters
Clusters
2
Nodes
Storage
Application
Cluster List
You can create up to 5 clusters and can add up to 20 nodes in each cluster.
Refresh
Create Kubernetes Cluster
Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands
Name
3
Manage
View Logs
Dashboard
Scale Cluster
More

Cluster Name/ID	Region	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
test [redacted]	China East 1 (Hangzhou)	VPC [redacted]	● Running	04/24/2018,09:58:28	1.9.3	Manage | View Logs Dashboard Scale Cluster | More

View the cluster operation information.



Time	Information
04/24/2018,13:55:38	c8def192cdbcc4af1a515d07fed184e44 | Start to client.DescribeTemplate
04/24/2018,13:55:35	c8def192cdbcc4af1a515d07fed184e44 | Start describeStackInfo
04/24/2018,11:27:38	c8def192cdbcc4af1a515d07fed184e44 | Start to client.DescribeTemplate
04/24/2018,11:27:36	c8def192cdbcc4af1a515d07fed184e44 | Start describeStackInfo
04/24/2018,11:27:08	c8def192cdbcc4af1a515d07fed184e44 | Start to client.DescribeTemplate
04/24/2018,11:27:06	c8def192cdbcc4af1a515d07fed184e44 | Start describeStackInfo
04/24/2018,11:26:55	c8def192cdbcc4af1a515d07fed184e44 | Start to client.DescribeTemplate
04/24/2018,11:26:54	c8def192cdbcc4af1a515d07fed184e44 | Start describeStackInfo
04/24/2018,11:25:02	c8def192cdbcc4af1a515d07fed184e44 | Start to client.DescribeTemplate
04/24/2018,11:25:00	c8def192cdbcc4af1a515d07fed184e44 | Start describeStackInfo
04/24/2018,10:16:42	c8def192cdbcc4af1a515d07fed184e44 | Set up k8s DNS configuration successfully
04/24/2018,10:15:36	c8def192cdbcc4af1a515d07fed184e44 | Start describeStackInfo
04/24/2018,10:15:36	c8def192cdbcc4af1a515d07fed184e44 | Stack CREATE completed successfully:
04/24/2018,10:15:34	c8def192cdbcc4af1a515d07fed184e44 | Start describeStackInfo

1.10.3 Use Log Service in a Kubernetes cluster

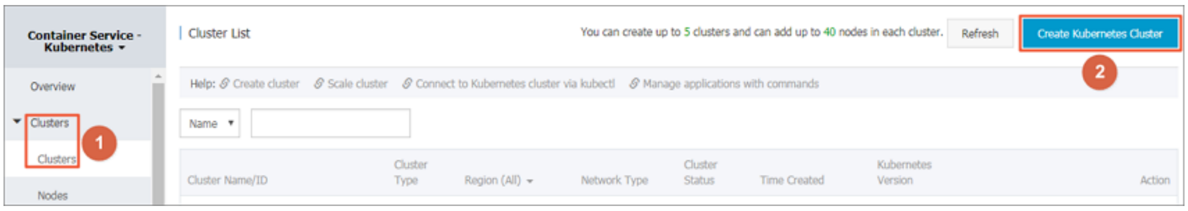
Log Service is integrated with Kubernetes clusters of Alibaba Cloud Container Service. To use Log Service in a Kubernetes cluster, enable Log Service when creating the cluster, and simply configure the application.

Prerequisites

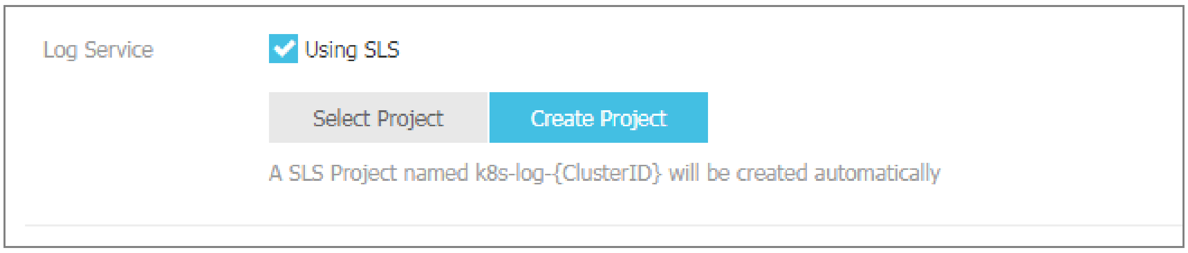
Make sure that you have enabled Alibaba Cloud *Log Service*.

Step 1 Install log components

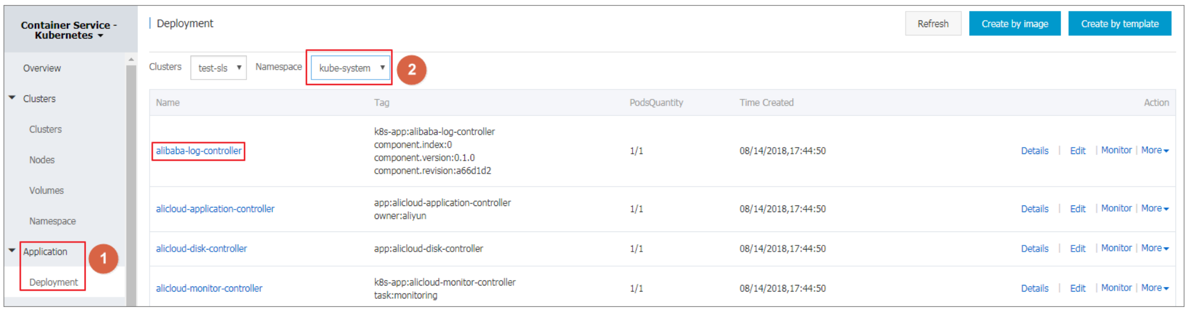
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Cluster** in the left-side navigation pane. Click **Create Kubernetes Cluster** in the upper-right corner of the page.


Container Service - Kubernetes
Overview
Clusters
Clusters
Nodes
1
Cluster List
You can create up to 5 clusters and can add up to 40 nodes in each cluster.
Refresh
Create Kubernetes Cluster
2
Help: Create cluster Scale cluster Connect to Kubernetes cluster via kubectl Manage applications with commands
Name
Cluster Name/ID
Cluster Type
Region (All)
Network Type
Cluster Status
Time Created
Kubernetes Version
Action

3. Click **Using SLS**. You can select an existing log project or create a new project directly. A project named`k8s-log-{ClusterID}` is created by default. For other cluster configuration parameters, see *Create a Kubernetes cluster*.


Log Service
Using SLS
Select Project
Create Project
A SLS Project named k8s-log-{ClusterID} will be created automatically

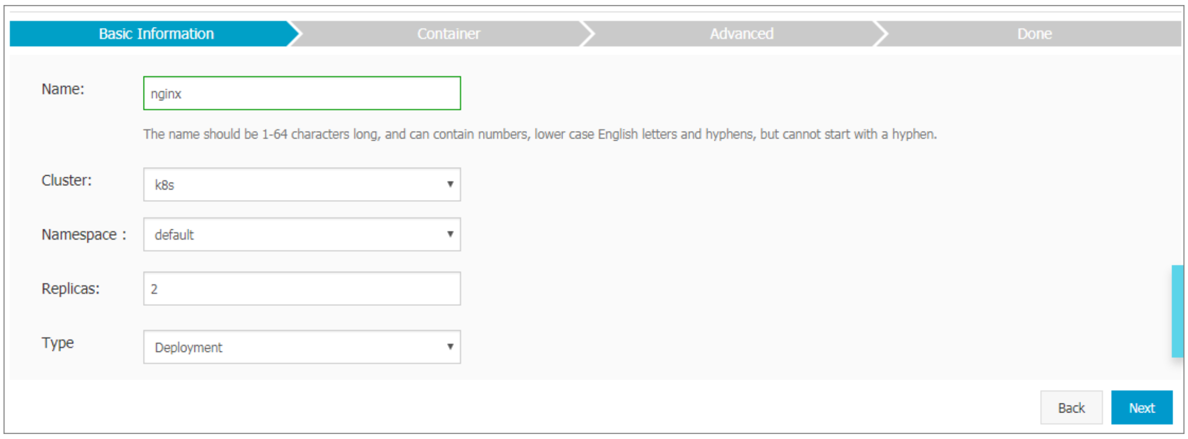
4. Click **Create** in the upper-right corner to start the deployment.
5. Click **Application** > **Deployment** in the left-side navigation pane, and select the target cluster and namespace. You can see that the alibaba-log-controller component is in running status.



Clusters: test-sls Namespace: kube-system

Name	Tag	PodsQuantity	Time Created	Action
alibaba-log-controller	k8s-app:alibaba-log-controller component.index:0 component.version:0.1.0 component.revision:a66d1d2	1/1	08/14/2018,17:44:50	Details | Edit | Monitor | More
alicloud-application-controller	app:alicloud-application-controller owner:aliyun	1/1	08/14/2018,17:44:50	Details | Edit | Monitor | More
alicloud-disk-controller	app:alicloud-disk-controller	1/1	08/14/2018,17:44:50	Details | Edit | Monitor | More
alicloud-monitor-controller	k8s-app:alicloud-monitor-controller task:monitoring	1/1	08/14/2018,17:44:50	Details | Edit | Monitor | More

6. Log on to the Log Service console, you can see that the log project is created.



Name: nginx

The name should be 1-64 characters long, and can contain numbers, lower case English letters and hyphens, but cannot start with a hyphen.

Cluster: k8s

Namespace : default

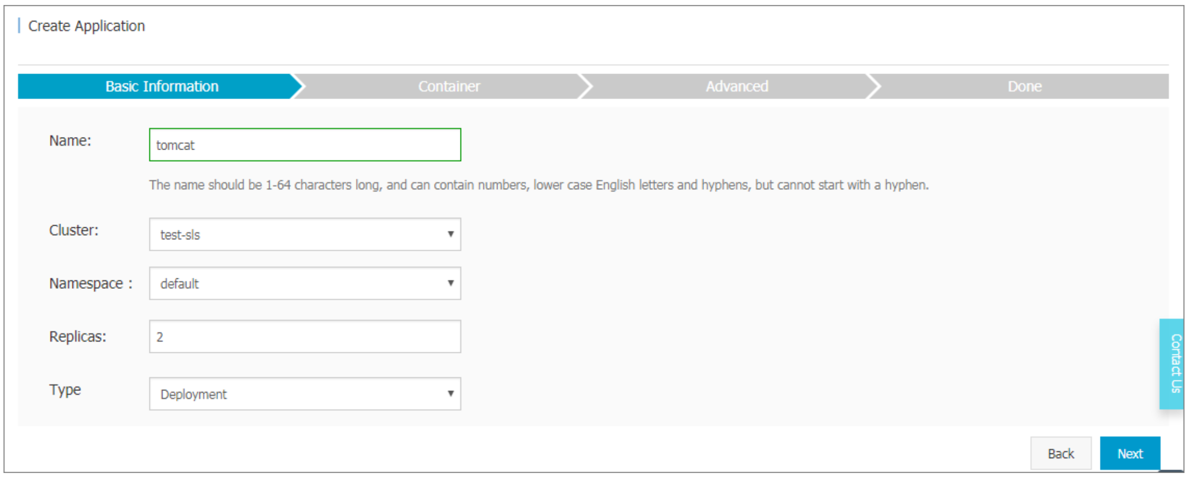
Replicas: 2

Type Deployment

Configure log collection on the deployment page

1. Log on to the *Container Service console*.

2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane. Click **Create by image** in the upper-right corner.
3. On the Basic Information page, configure basic parameters of the application, including the application name, cluster, and namespace. Click**Nest**.



Name: tomcat

The name should be 1-64 characters long, and can contain numbers, lower case English letters and hyphens, but cannot start with a hyphen.

Cluster: test-sls

Namespace : default

Replicas: 2

Type Deployment

4. On the Container page, first configure the image name, image version, and the number of containers. In this example, create a tomcat application to test the collection of container standard output logs, and the container text logs.

Note:

Tomcat images are suitable for the demonstration in this example because they belong to the minority container images that use both stdout and file logs.

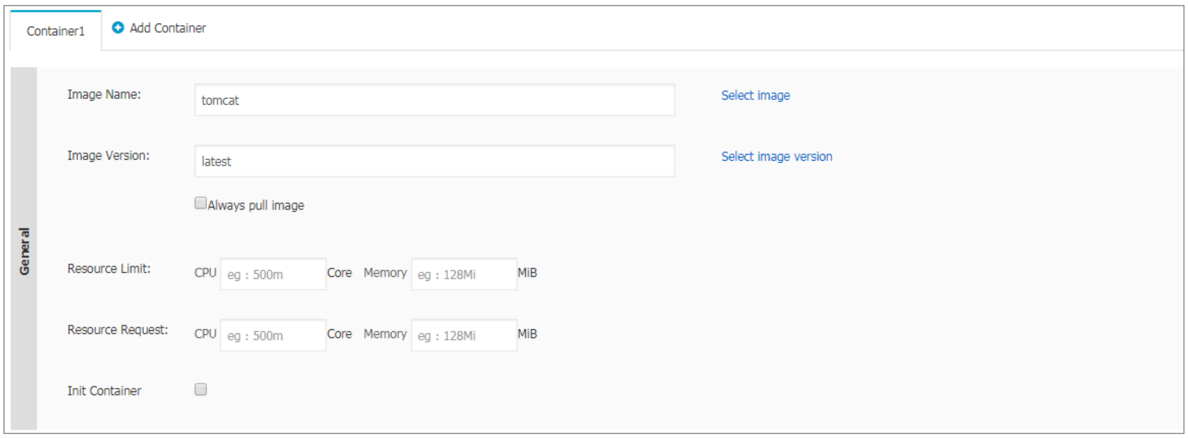


Image Name: tomcat

Image Version: latest

Always pull image

Resource Limit: CPU eg : 500m Core Memory eg : 128Mi MiB

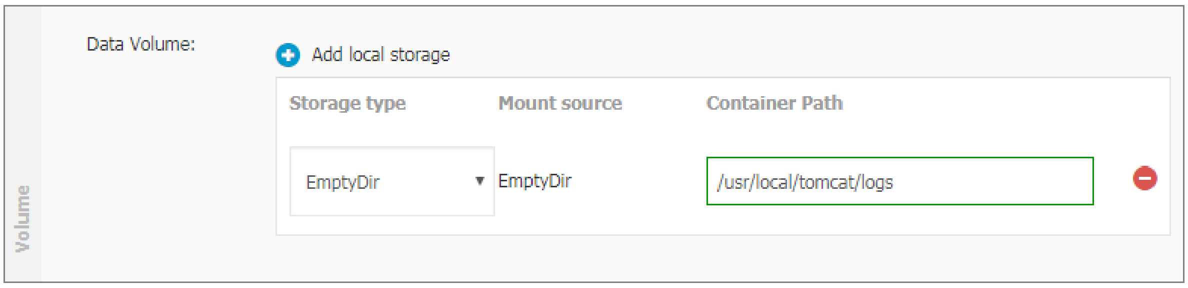
Resource Request: CPU eg : 500m Core Memory eg : 128Mi MiB

Init Container

5. Configure a data volume to mount the log directory in the container to the temporary directory in which text logs are save.

Note:

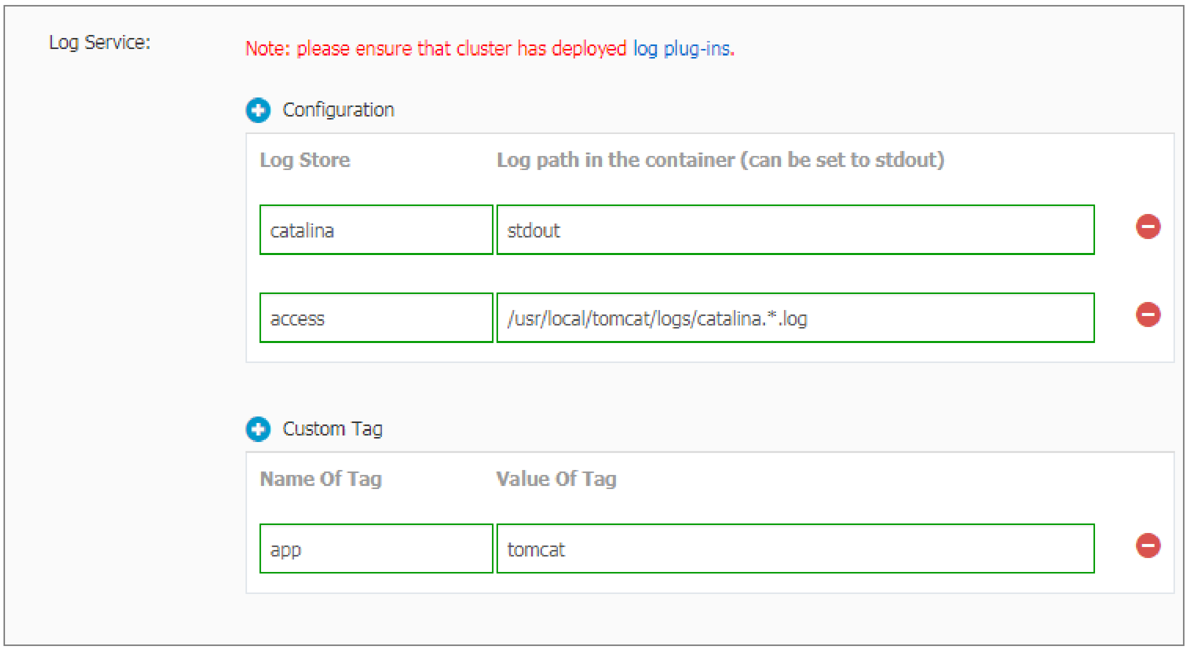
You can also use other storage to save container text logs, such as cloud disks, NAS, and other types of cloud storage.


Volume

Data Volume: Add local storage

Storage type	Mount source	Container Path
EmptyDir	EmptyDir	/usr/local/tomcat/logs

6. Configure Log Service. You can configure Logstores and set custom tag settings.


Log Service:

Note: please ensure that cluster has deployed log plug-ins.

Configuration

Log Store	Log path in the container (can be set to stdout)
catalina	stdout
access	/usr/local/tomcat/logs/catalina.*.log

Custom Tag

Name Of Tag	Value Of Tag
app	tomcat

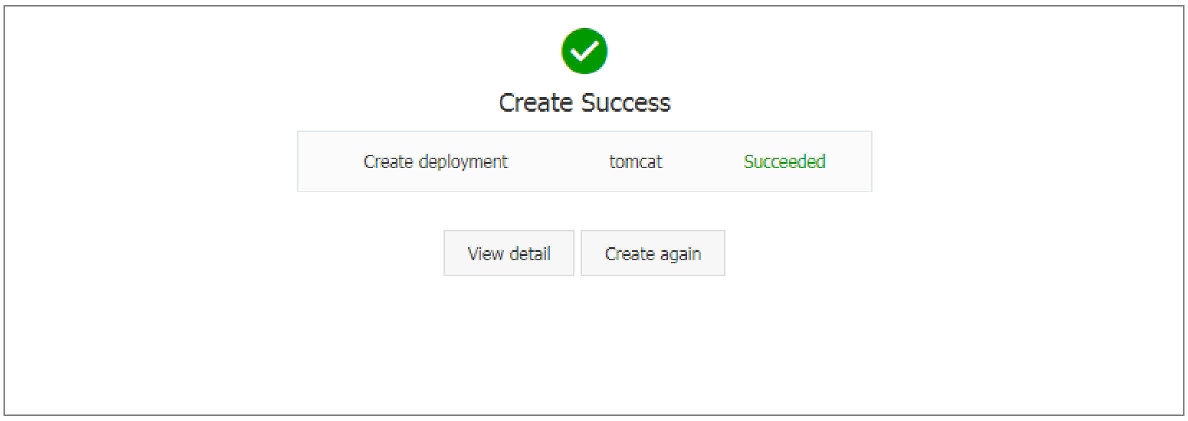
Configure Log Service as follows:

- **Log Store**: The Logstore generated in Log Service which is use 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 of a specified path in containers and supports wildcard characters. In this example, the log path specified in the preceding figure indicates to collect text logs that meet the name rule of /usr/local/tomcat/logs/catalina. *.log.

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 convenience to log analysis such as log statistics and filter.

7. Click **Next** after completing the configurations.

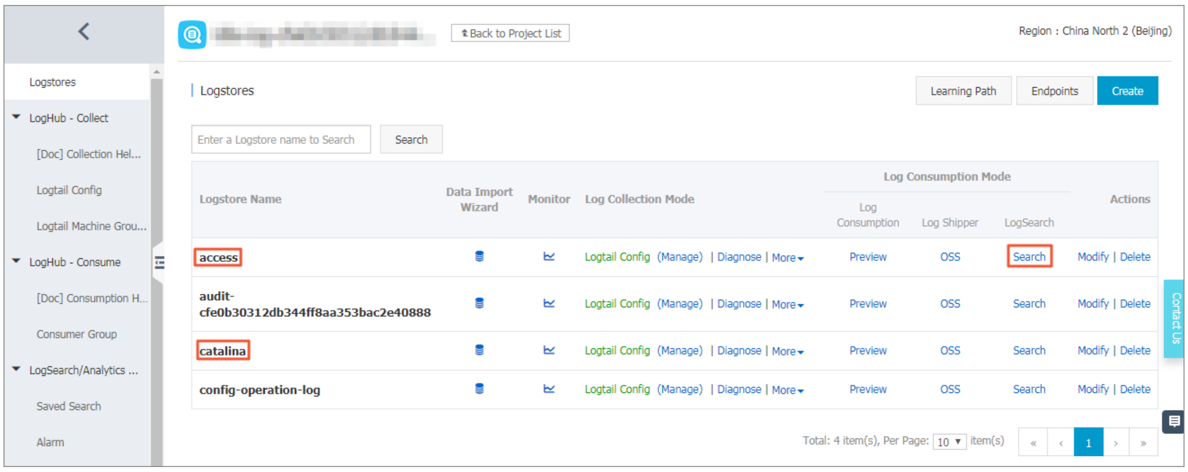
8. On the Access Control page, click**Create** directly.



Create Success

Create deployment	tomcat	Succeeded

9. Enter the Log service console, you can see that the created Logstore is displayed in the corresponding log project. Select a logstore and click **Search**.



Logstore Name	Data Import Wizard	Monitor	Log Collection Mode	Log Consumption Mode			Actions
				Log Consumption	Log Shipper	LogSearch	
access			Logtail Config (Manage) | Diagnose | More	Preview	OSS	Search	Modify | Delete
audit-cfe0b30312db344ff8aa353bac2e40888			Logtail Config (Manage) | Diagnose | More	Preview	OSS	Search	Modify | Delete
catalina			Logtail Config (Manage) | Diagnose | More	Preview	OSS	Search	Modify | Delete
config-operation-log			Logtail Config (Manage) | Diagnose | More	Preview	OSS	Search	Modify | Delete

Total: 4 item(s), Per Page: 10 item(s) « ‹ 1 › »

10.The page jumps to the log list page. When you select log query, you can see that each log is tagged with the `app=tomcat` tag.

1.10.4 A solution to log collection problems of Kubernetes clusters 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.

Currently, log-pilot is completely open-source in GitHub. The project address is *https://github.com/AliyunContainerService/log-pilot*. You can know more implementation principles about it.

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_logs_$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_logs_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 regxp. 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

You have activated Container Service and created a Kubernetes cluster. In this example, create a Kubernetes cluster in China East 1 (Hangzhou).

Step 1 Deploy Elasticsearch

1. Connect to your Kubernetes cluster. For more information, see *#unique_31* or *#unique_134*.
2. Deploy the resource object related to Elasticsearch first. Then, enter the following orchestration template. This orchestration template includes an elasticsearch-api service, an elasticsearch-discovery service, and a status set of Elasticsearch. All of these objects are deployed under the namespace kube-system.

```
kubectl apply -f https://acs-logging.oss-cn-hangzhou.aliyuncs.com/
elasticsearch.yml
```

3. After the successful deployment, corresponding objects are under the namespace kube-system. Run the following commands to check the running status:

```
$ kubectl get svc,StatefulSet -n=kube-system
 NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
 svc/elasticsearch-api ClusterIP 172.21.5.134 <none> 9200/TCP 22h
 svc/elasticsearch-discovery ClusterIP 172.21.13.91 <none> 9300/TCP
22h
 ...
 NAME DESIRED CURRENT AGE
 statefulsets/elasticsearch 3 3 22h
```

Step 2 Deploy log-pilot and the Kibana service

1. Deploy the log-pilot log collection tool. The orchestration template is as follows:

```
kubectl apply -f https://acs-logging.oss-cn-hangzhou.aliyuncs.com/
log-pilot.yml
```

2. Deploy the Kibana service. The sample orchestration template contains a service and a deployment.

```
kubectl apply -f https://acs-logging.oss-cn-hangzhou.aliyuncs.com/
kibana.yml
```

Step 3 Deploy the test application Tomcat

After deploying the log tool set of Elasticsearch + log-pilot + Kibana, deploy a test application Tomcat to test whether or not logs can be successfully collected, indexed, and displayed.

The orchestration template is as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: tomcat
  namespace: default
  labels:
    name: tomcat
spec:
  containers:
  - image: tomcat
    name: tomcat-test
    volumeMounts:
    - mountPath: /usr/local/tomcat/logs
      name: accesslogs
    env:
     - name: aliyun_logs_catalina
       value: "stdout" ##Collect standard output logs.
     - name: aliyun_logs_access
       value: "/usr/local/tomcat/logs/catalina. *.log" ## Collect log
files within the container.
  volumes:
    - name: accesslogs
      emptyDir: {}
```

The Tomcat image is a Docker image that both uses stdout and file logs. In the preceding orchestration, the log collection configuration file is dynamically generated by defining the environment variable in the pod. See the following descriptions for the environment variable:

- `aliyun_logs_catalina=stdout` indicates to collect stdout logs from the container.

- `aliyun_logs_access=/usr/local/tomcat/logs/catalina. *.log` indicates to collect all the log files whose name matches *`catalina. *.log`* under the directory *`/usr/local/tomcat/logs/`* from the container.

In the Elasticsearch scenario of this solution, the `$name` in the environment variable indicates index. In this example, `$name` is catalina and access.

Step 4 Expose the Kibana service to Internet

The Kibana service deployed in the preceding section is of the NodePort type, which cannot be accessed from the Internet by default. Therefore, create an Ingress in this document to access the Kibana service from Internet and test whether or not logs are successfully indexed and displayed.

1. Create an Ingress to access the Kibana service from Internet. In this example, use the simple routing service to create an Ingress. For more information, see *#unique_135*. The orchestration template of the Ingress is as follows:

```
apiVersion: extensions/v1beta1
 kind: Ingress
 metadata:
   name: kibana-ingress
   namespace: kube-system #Make sure the namespace is the same as
that of the Kibana service.
 spec:
   rules:
   - http:
       paths:
       - path: /
         backend:
           serviceName: kibana #Enter the name of the Kibana service
.
           servicePort: 80 #Enter the port exposed by the Kibana
service.
```

2. After the Ingress is successfully created, run the following commands to obtain the access address of the Ingress:

```
$ kubectl get ingress -n=kube-system
 NAME HOSTS ADDRESS PORTS AGE
 shared-dns * 120.55.150.30 80 5m
```

3. Access the address in the browser as follows.

4. Click **Management** in the left-side navigation pane. Then, click **Index Patterns** > **Create Index Pattern**. The detailed index name is the `$name` variable suffixed with a time string. You can create an index pattern by using the wildcard `*`. In this example, use `$name*` to create an index pattern.

You can also run the following commands to enter the corresponding pod of Elasticsearch and list all the indexes of Elasticsearch:

```
$ kubectl get pods -n=kube-system #Find the corresponding pod of
Elasticsearch.
    ...
$ kubectl exec -it elasticsearch-1 bash #Enter a pod of Elasticsea
rch.
  ...
$ curl 'localhost:9200/_cat/indices? v' ## List all the indexes.
 health status index uuid pri rep docs.count docs.deleted store.size
 pri.store.size
 green open .kibana x06jj19PS4Cim6Ajo51PWg 1 1 4 0 53.6kb 26.8kb
 green open access-2018.03.19 txd3tG-NR6-guqmMEKKzEw 5 1 143 0 823.
5kb 411.7kb
 green open catalina-2018.03.19 ZgtWd16FQ7qqJNNWXxFPcQ 5 1 143 0 915
.5kb 457.5kb
```

5. After successfully creating the indexes, click **Discover** in the left-side navigation pane, select the created index and the corresponding time range, and then enter the related field in the search box to query logs.

Then, you have successfully tested the solution to log collection problems of Alibaba Cloud Kubernetes clusters based on log-pilot, Elasticsearch, and Kibana. By using this solution, you can deal with requirements for logs of distributed Kubernetes clusters effectively, improve the Operation and Maintenance and operational efficiencies, and guarantee the continuous and stable running of the system.

1.10.5 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 configuring 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).

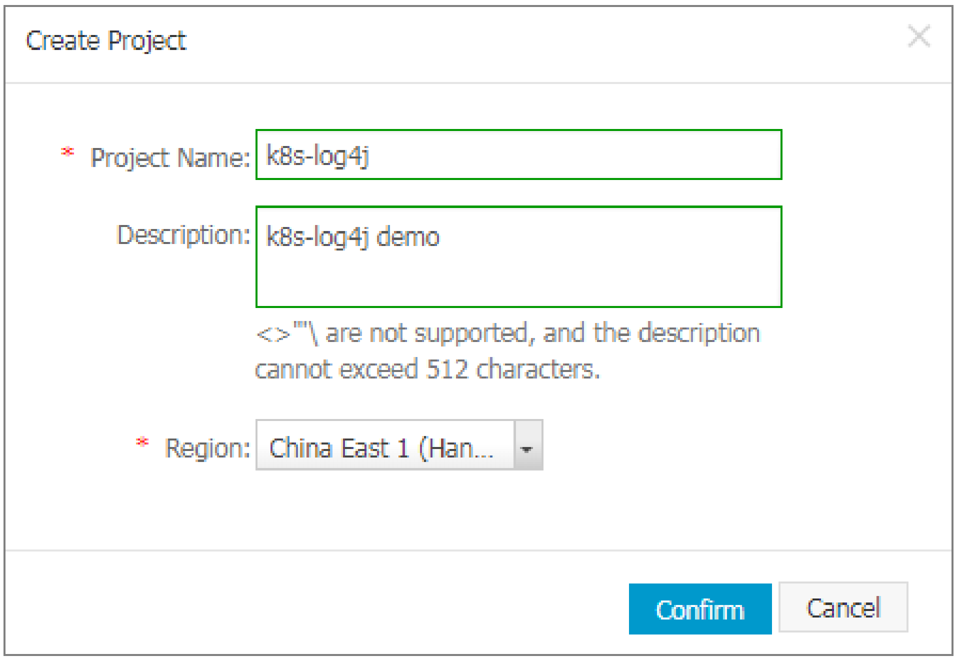
- Enable AccessKey or Resource Access Management (RAM). Make sure you have sufficient access permissions. Use the AccessKey in this example.

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.



Create Project

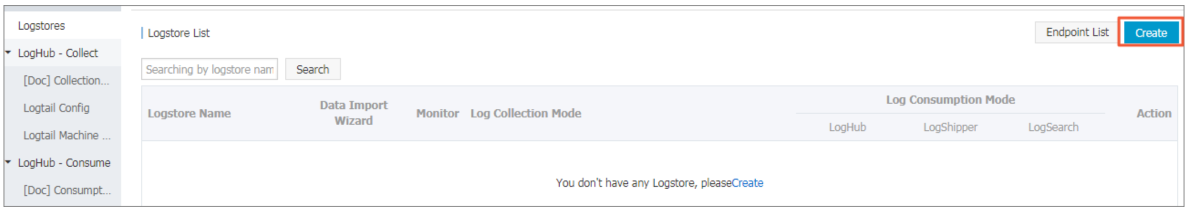
* Project Name: k8s-log4j

Description: k8s-log4j demo

<>"'\ are not supported, and the description cannot exceed 512 characters.

* Region: China East 1 (Han...

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.

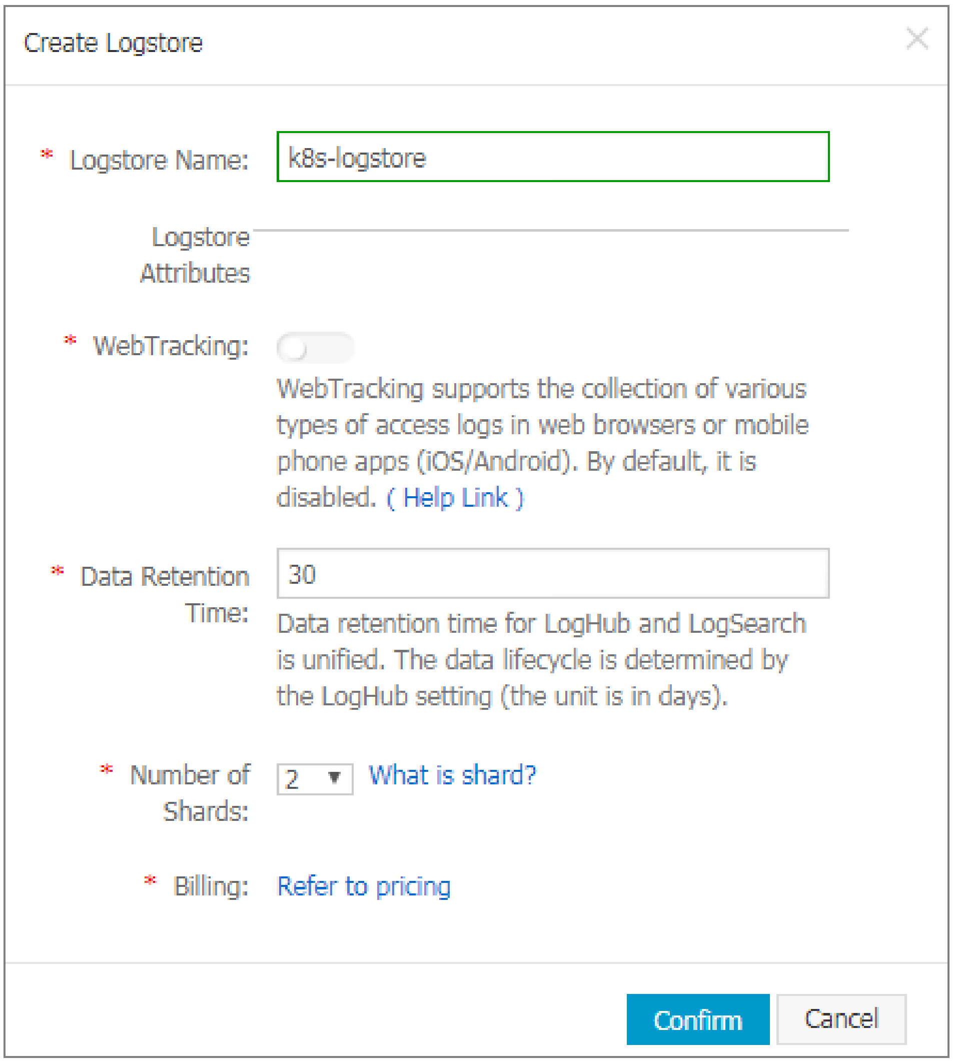


Logstore Name	Data Import Wizard	Monitor	Log Collection Mode	Log Consumption Mode			Action
				LogHub	LogShipper	LogSearch	

You don't have any Logstore, pleaseCreate

5. Complete the configurations and then click **Confirm**.

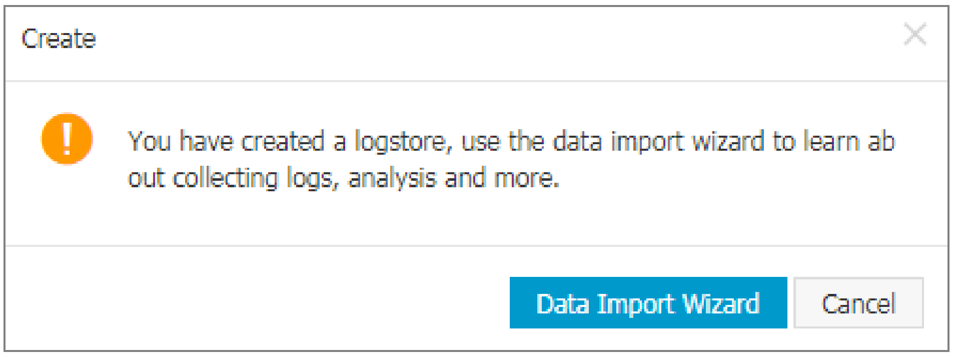
   In this example, create a Logstore named k8s-logstore.



Create Logstore

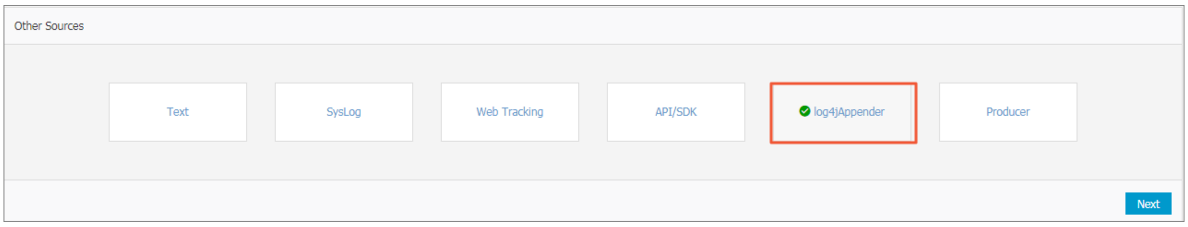
* Logstore Name:	k8s-logstore
Logstore Attributes	
* WebTracking:	WebTracking supports the collection of various types of access logs in web browsers or mobile phone apps (iOS/Android). By default, it is disabled. (Help Link)
* Data Retention Time:	30 Data retention time for LogHub and LogSearch is unified. The data lifecycle is determined by the LogHub setting (the unit is in days).
* Number of Shards:	2 What is shard?
* Billing:	Refer to pricing

6. Then, a dialog box asking you to use the data import wizard appears.


Create
You have created a logstore, use the data import wizard to learn ab out collecting logs, analysis and more.
Data Import Wizard
Cancel

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 *Access Kubernetes clusters by using SSH* or *Connect to a Kubernetes cluster by using kubectl*.

2. Obtain the *demo-deployment.yaml* file and configure the environment variable `JAVA_OPTS` to collect logs from the Kubernetes cluster.

   The sample orchestration 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:
    matchLabels:
      app: log4j-appender
  template:
    metadata:
```

```
    labels:
      app: log4j-appender
  spec:
    containers:
    - name: log4j-appender-demo-spring-boot
      image: registry.cn-hangzhou.aliyuncs.com/jaegertracing/log4j-
appender-demo-spring-boot:0.0.2
      env:
      - name: JAVA_OPTS ##Note
        value: "-Dproject={your_project} -Dlogstore={your_logstore
} -Dendpoint={your_endpoint} -Daccess_key_id={your_access_key_id} -
Daccess_key={your_access_key_secret}"
      ports:
      - containerPort: 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_key_id`: Your AccessKey ID.
- `-Daccess_key`: 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 LoadBalancer 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_SERVICE_IP` is the `EXTERNAL-IP`.

Note:

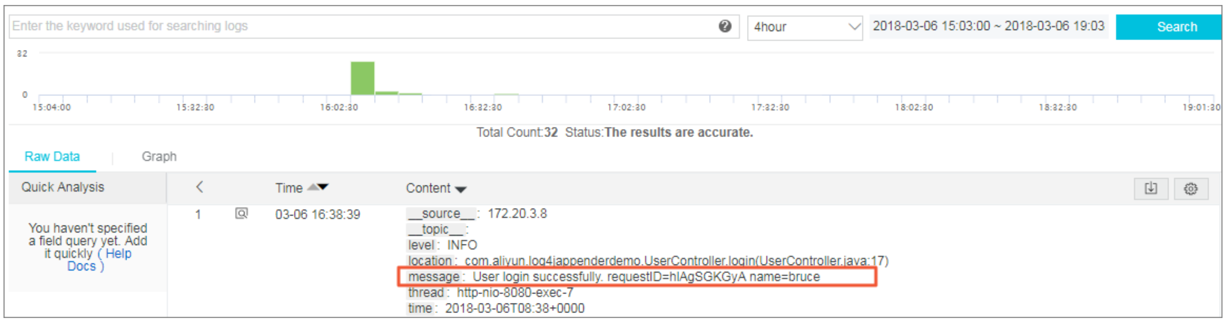
See *GitHub log4j-appender-demo* to view the complete collection of APIs.

```
curl http://${K8S_SERVICE_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.



```
1    03-06 16:38:39    __source__: 172.20.3.8
                       __topic__:
                       level: INFO
                       location: com.aliyun.log4jappenderdemo.UserController.login(UserController.java:17)
                       message: User login successfully. requestID=hlAgSGKGyA name=bruce
                       thread: http-nio-8080-exec-7
                       time: 2018-03-06T08:38+0000
```

The output content of the log corresponds to the preceding command. This example demonstrat es 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.

1.11 Monitoring

1.11.1 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.

Background information

A monitoring system monitors the following two types of objects:

- Resource, which is 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, which includes internal application metrics. 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, specify monitoring methods for them in the configuration file.

To monitor dynamic resource entities, we recommend that you deploy the Prometheus monitoring system.

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*.

Deploy the Prometheus monitoring system

1. Run the following command to download the prometheus-operator code:

```
git clone https://github.com/AliyunContainerService/prometheus-
operator
```

2. Run the following command to deploy the Prometheus monitoring system:

```
cd prometheus-operator/contrib/kube-prometheus
```

```
kubectl apply -f manifests
```

3. Run the following command to set the Prometheus access:

```
kubectl --namespace monitoring port-forward svc/prometheus-k8s 9090
```

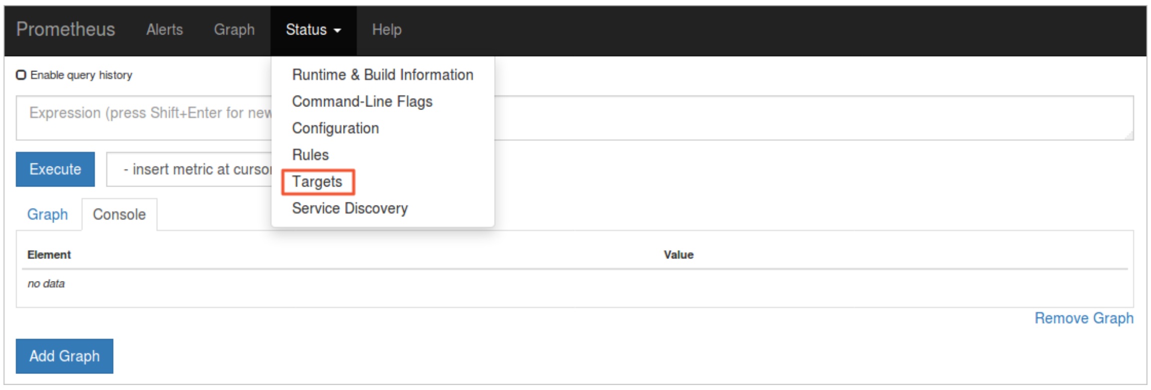
4. View the deployment result

   a. To view Prometheus, access`localhost:9090` in a browser.

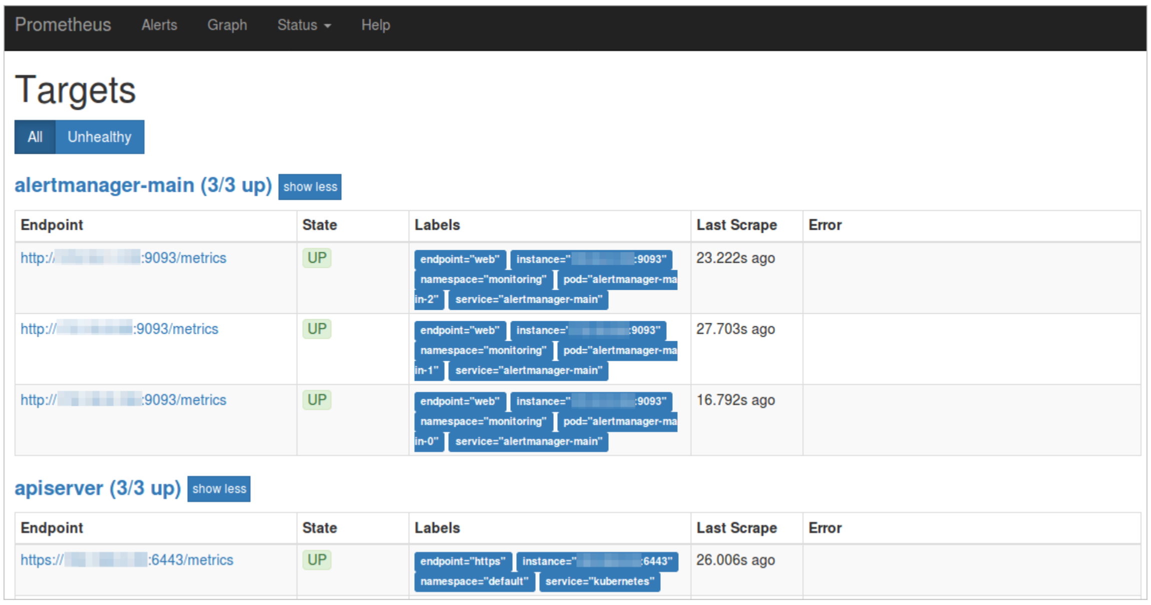
   Note:

   By default, Prometheus cannot be accessed through the Internet. You must use your local proxy to access it.

   b. 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.



Targets

alertmanager-main (3/3 up)

Endpoint	State	Labels	Last Scrape	Error
http://[redacted]:9093/metrics	UP	endpoint="web" instance="[redacted]:9093" namespace="monitoring" pod="alertmanager-main-2" service="alertmanager-main"	23.222s ago	
http://[redacted]:9093/metrics	UP	endpoint="web" instance="[redacted]:9093" namespace="monitoring" pod="alertmanager-main-1" service="alertmanager-main"	27.703s ago	
http://[redacted]:9093/metrics	UP	endpoint="web" instance="[redacted]:9093" namespace="monitoring" pod="alertmanager-main-0" service="alertmanager-main"	16.792s ago	

apiserver (3/3 up)

Endpoint	State	Labels	Last Scrape	Error
https://[redacted]:6443/metrics	UP	endpoint="https" instance="[redacted]:6443" namespace="default" service="kubernetes"	26.006s ago	

View and display data aggregation

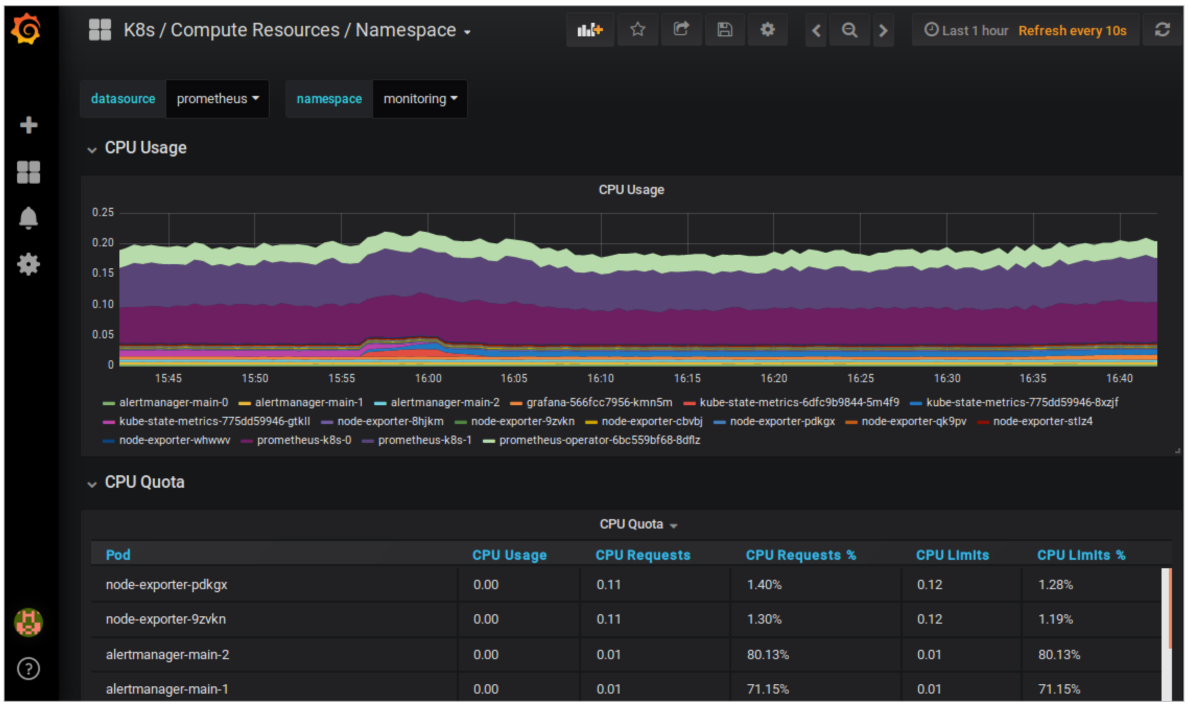
1. Run the following command to access Grafana:

```
kubectl --namespace monitoring port-forward svc/grafana 3000
```

2. Access `localhost:3000` in a browser and then select a dashboard to view data aggregation.

Note:

The default user name and password are both admin.



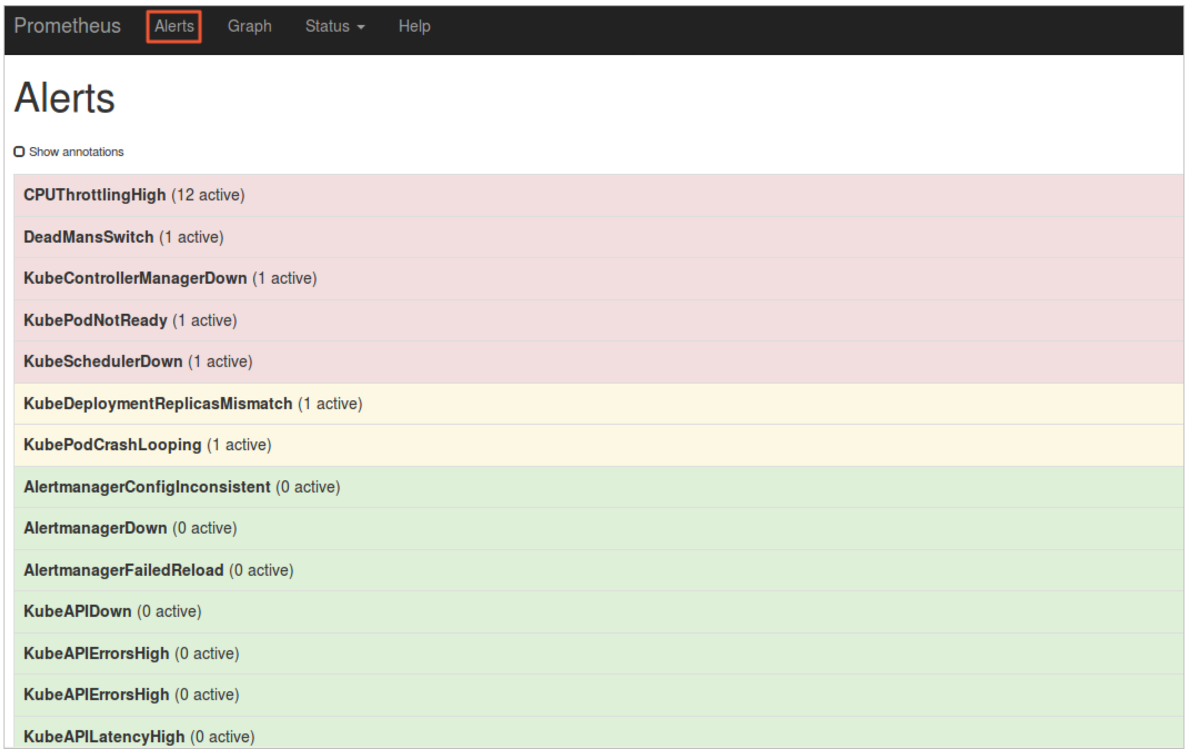
Pod	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
node-exporter-pdkgx	0.00	0.11	1.40%	0.12	1.28%
node-exporter-9zvkn	0.00	0.11	1.30%	0.12	1.19%
alertmanager-main-2	0.00	0.01	80.13%	0.01	80.13%
alertmanager-main-1	0.00	0.01	71.15%	0.01	71.15%

View alerting rules and set alert silencing

- **View alerting rules**

  Open `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 normal status.



Alerts

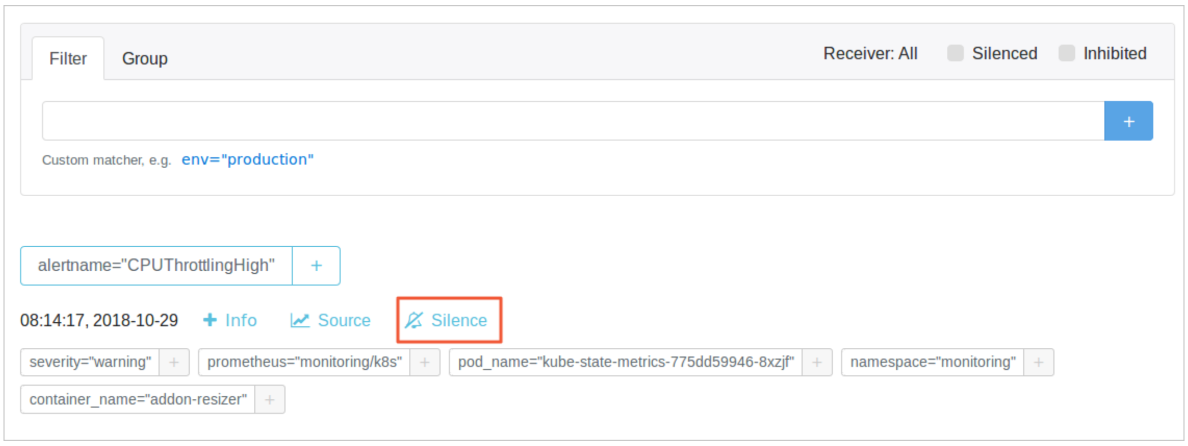
Show annotations

CPUThrottlingHigh (12 active)
DeadMansSwitch (1 active)
KubeControllerManagerDown (1 active)
KubePodNotReady (1 active)
KubeSchedulerDown (1 active)
KubeDeploymentReplicasMismatch (1 active)
KubePodCrashLooping (1 active)
AlertmanagerConfigInconsistent (0 active)
AlertmanagerDown (0 active)
AlertmanagerFailedReload (0 active)
KubeAPIDown (0 active)
KubeAPIErrorsHigh (0 active)
KubeAPIErrorsHigh (0 active)
KubeAPILatencyHigh (0 active)

- **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/alertmanager-main
9093
```



Custom matcher, e.g. env="production"

alertname="CPUThrottlingHigh"

08:14:17, 2018-10-29

severity="warning" prometheus="monitoring/k8s" pod_name="kube-state-metrics-775dd59946-8xzjf" namespace="monitoring" container_name="addon-resizer"

1.11.2 Group-based monitoring and alarms

Alibaba Cloud Container Service is interoperable with CloudMonitor to enable group-based monitoring and alarms.

Prerequisites

- *Create a Kubernetes cluster* if you do not have one.
- The Kubernetes version must be 1.8.4 or later. Otherwise, you must first upgrade the cluster.

Context

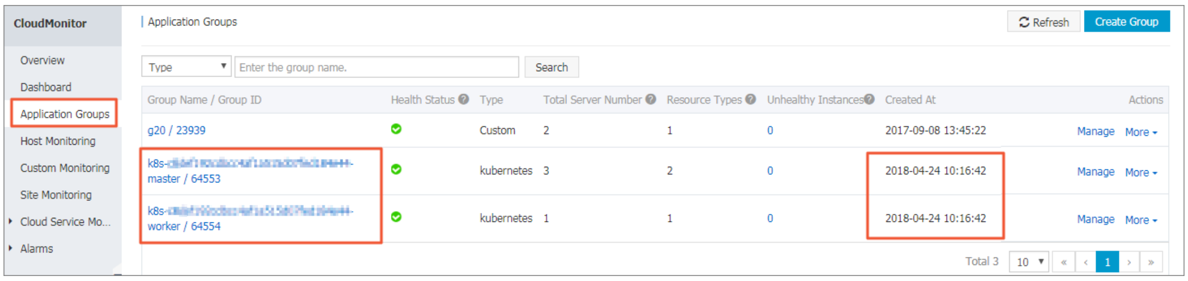
In the Operation & Maintenance (O&M) of IT infrastructure, monitoring and alarms facilitate daily O &M, system monitoring, troubleshooting, and debugging, and guarantee the reliability and security of O&M.

The traditional container monitoring solution that uses a statically configured monitoring agent or a centralized server for monitoring and alarms may not be suitable for the Kubernetes scenario because it can cause some problems. For example, the information required to identify the monitoring objects is missing because containers are mostly scheduled in the resource pool whereas the monitoring agent is deployed on the host. Also, containers have shorter lives than applications. The monitoring and alarm rules, and such monitoring data as ReplicaSet and Deployment for a single container cannot be used for the corresponding application.

Alibaba Cloud Container Service for Kubernetes is deeply integrated with CloudMonitor to use application groups to unify the monitoring objects and metrics. In addition, CloudMonitor of Alibaba Cloud is equipped with many functions and custom tools, which provide you with the best practice to monitor your Kubernetes resources and manage the alarms.

Procedure

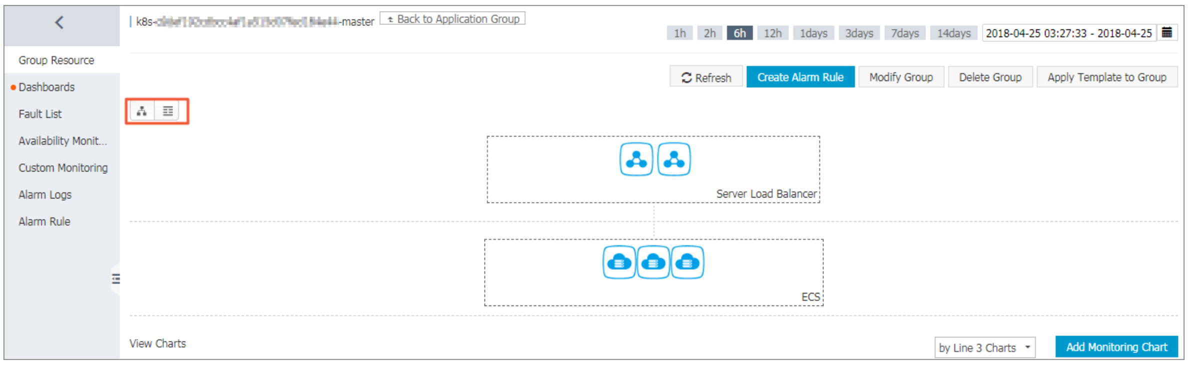
1. Log on to the *CloudMonitor console*.
2. In the left-side navigation pane, click **Application Groups**. The Kubernetes groups with cluster IDs are displayed.



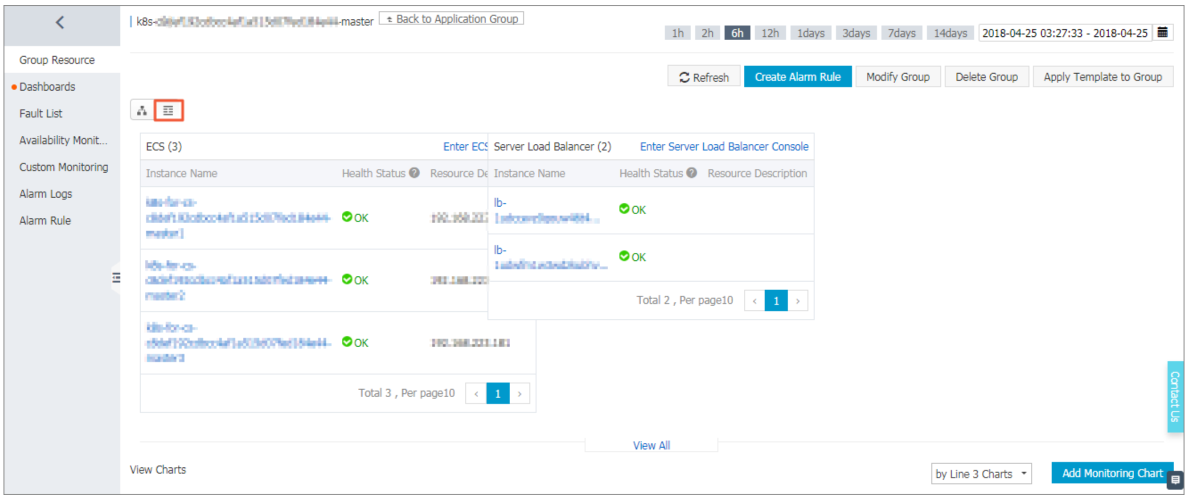
Group Name / Group ID	Health Status	Type	Total Server Number	Resource Types	Unhealthy Instances	Created At	Actions
g20 / 23939	✓	Custom	2	1	0	2017-09-08 13:45:22	Manage More
k8s-[redacted]-master / 64553	✓	kubernetes	3	2	0	2018-04-24 10:16:42	Manage More
k8s-[redacted]-worker / 64554	✓	kubernetes	1	1	0	2018-04-24 10:16:42	Manage More

3. Click the **Group Name** to go to the group details page. You can view the resources contained in the group. For example, in a Master group of Kubernetes, you can see such resources as Elastic Compute Service (ECS) instances and Server Load Balancer (SLB) instances.

   Kubernetes has two types of nodes: Worker nodes and Master nodes. Master nodes generally contain management and control applications and the resources are required to be highly robust. Worker nodes are generally responsible for scheduling pods and the overall requirement on the resources focuses on scheduling capability. When you create a group, Container Service automatically creates two resource groups, a Master group and a Worker group. The Master group includes the Master nodes and the related SLB instances. The Worker group includes all the Worker nodes.



4. You can view the details of other cloud products, such as SLB, in the group.



ECS (3) Enter ECS

Instance Name	Health Status	Resource De
[illegible]	OK	[illegible]
[illegible]	OK	[illegible]
[illegible]	OK	[illegible]

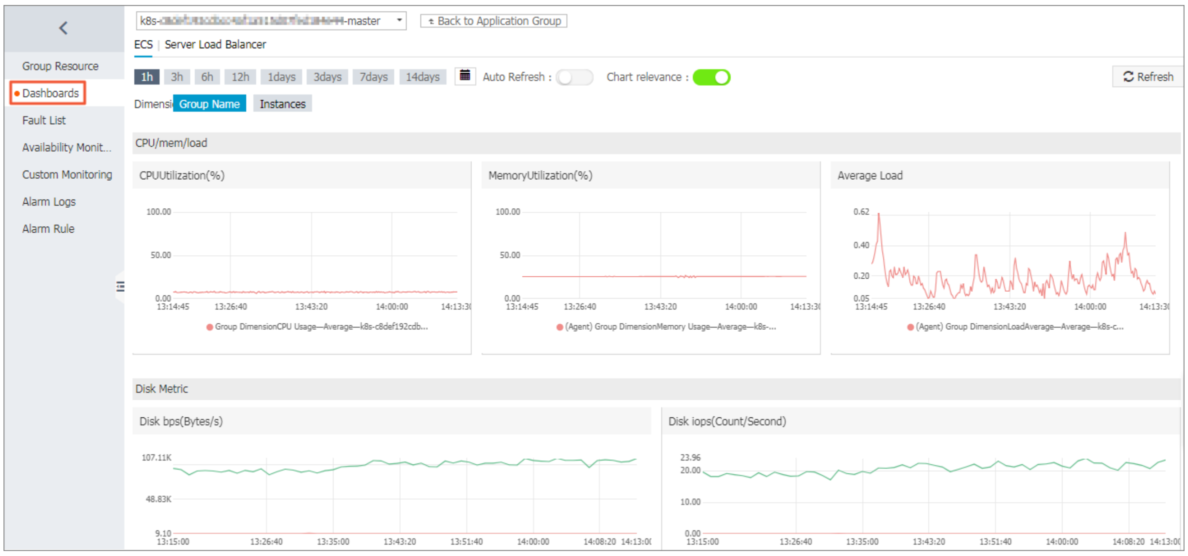
Total 3 , Per page10 ‹ 1 ›

Server Load Balancer (2) Enter Server Load Balancer Console

Instance Name	Health Status	Resource Description
lb-[illegible]	OK	
lb-[illegible]	OK	

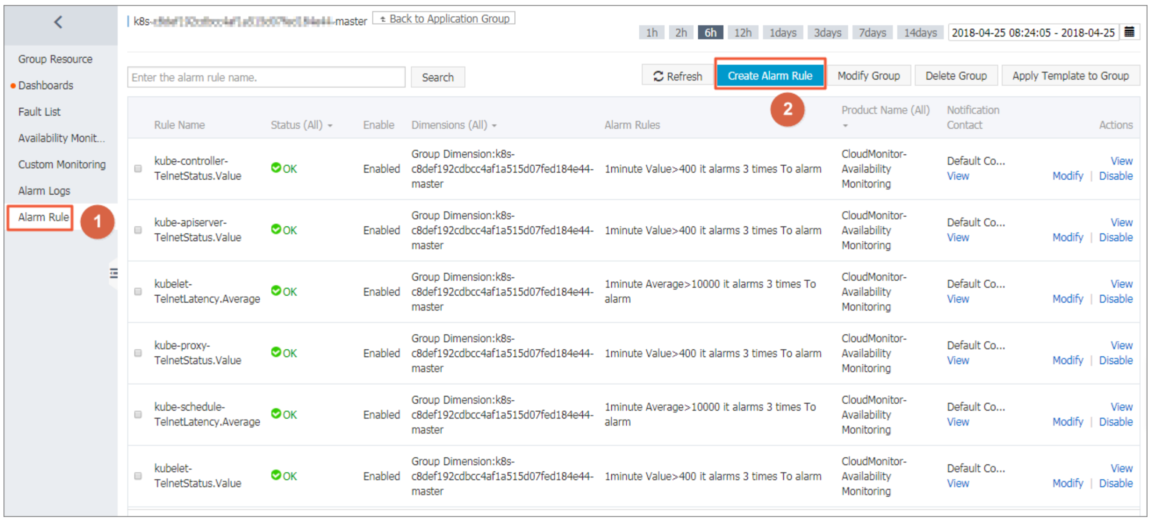
Total 2 , Per page10 ‹ 1 ›

5. In the left-side navigation pane, click **Dashboards** to view the detailed monitoring metrics of each cloud product in the group.



6. In the left-side navigation pane, click **Alarm Rule**. A list of existing alarm rules in the group is displayed. By default, the health of the core components of all nodes in the Mater group is checked.

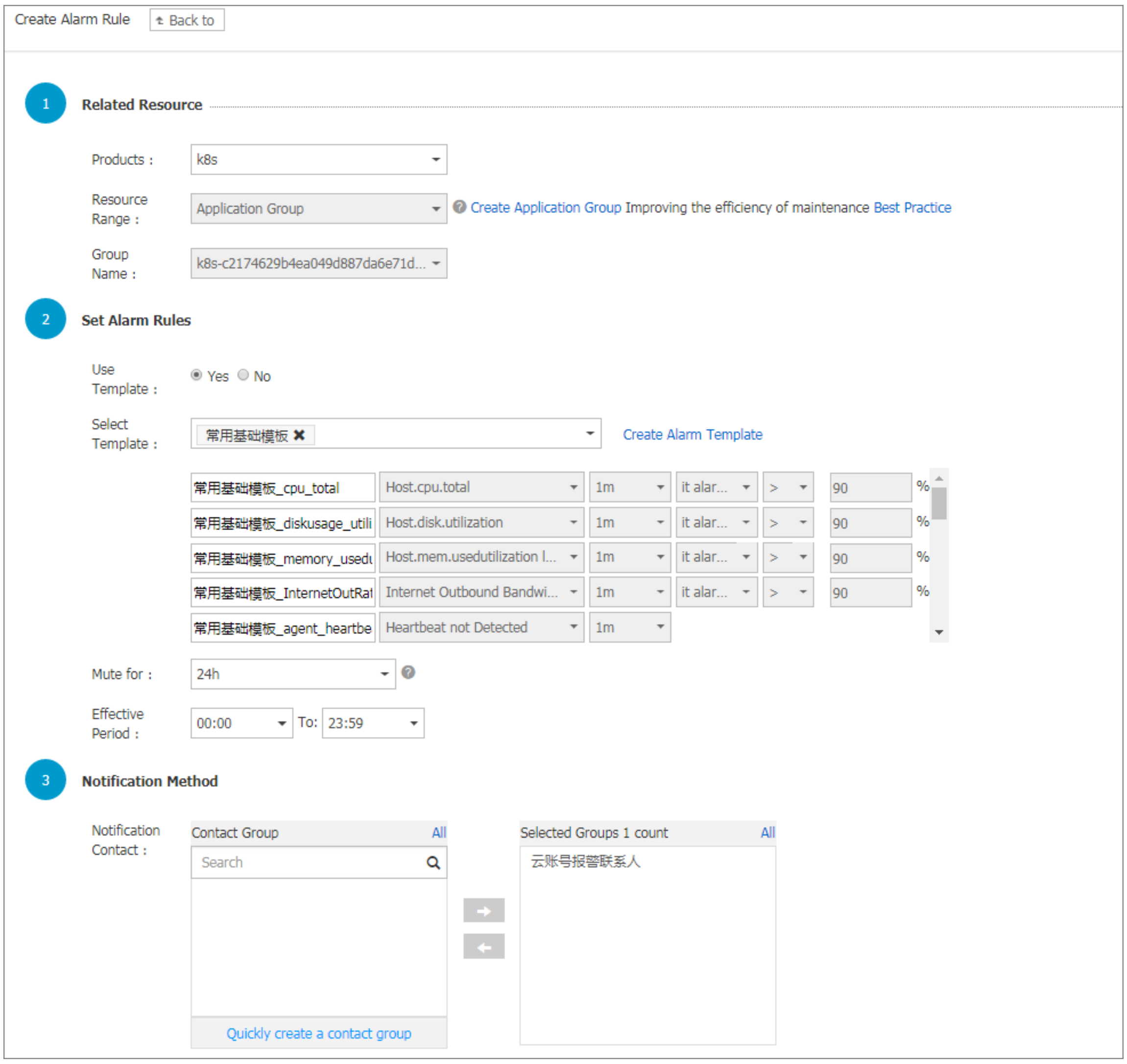
   1. Click **Create Alarm Rule** to create an alarm rule for the group according to your business requirements.



Rule Name	Status (All)	Enable	Dimensions (All)	Alarm Rules	Product Name (All)	Notification Contact	Actions
kube-controller-TelnetStatus.Value	OK	Enabled	Group Dimension:k8s-c8def192cdbcc4af1a515d07fed184e44-master	1minute Value>400 it alarms 3 times To alarm	CloudMonitor-Availability Monitoring	Default Co... View	View Modify | Disable
kube-apiserver-TelnetStatus.Value	OK	Enabled	Group Dimension:k8s-c8def192cdbcc4af1a515d07fed184e44-master	1minute Value>400 it alarms 3 times To alarm	CloudMonitor-Availability Monitoring	Default Co... View	View Modify | Disable
kubelet-TelnetLatency.Average	OK	Enabled	Group Dimension:k8s-c8def192cdbcc4af1a515d07fed184e44-master	1minute Average>10000 it alarms 3 times To alarm	CloudMonitor-Availability Monitoring	Default Co... View	View Modify | Disable
kube-proxy-TelnetStatus.Value	OK	Enabled	Group Dimension:k8s-c8def192cdbcc4af1a515d07fed184e44-master	1minute Value>400 it alarms 3 times To alarm	CloudMonitor-Availability Monitoring	Default Co... View	View Modify | Disable
kube-schedule-TelnetLatency.Average	OK	Enabled	Group Dimension:k8s-c8def192cdbcc4af1a515d07fed184e44-master	1minute Average>10000 it alarms 3 times To alarm	CloudMonitor-Availability Monitoring	Default Co... View	View Modify | Disable
kubelet-TelnetStatus.Value	OK	Enabled	Group Dimension:k8s-c8def192cdbcc4af1a515d07fed184e44-master	1minute Value>400 it alarms 3 times To alarm	CloudMonitor-Availability Monitoring	Default Co... View	View Modify | Disable

   2. On the displayed **Create Alarm Rule** page, set the alarm rules.

      - Select the related resource, such as ECS.
      - Select whether to use a template to create the alarm rule. If yes, select an alarm template from the **Select Template** drop-down list. You can also click **Create Alarm Template** to create a new custom alarm template. For more information, see .
      - Set the notification method. For example, you can know the Kubernetes cluster status through DingTalk, email, and SMS.



Create Alarm Rule ↥ Back to

1 **Related Resource**

Products : k8s

Resource Range : Application Group — Create Application Group Improving the efficiency of maintenance Best Practice

Group Name : k8s-c2174629b4ea049d887da6e71d...

2 **Set Alarm Rules**

Use Template : [✓ Yes] No

Select Template : 常用基础模板 ✖ — Create Alarm Template

常用基础模板_cpu_total	Host.cpu.total	1m	it alar...	>	90	%
常用基础模板_diskusage_utili	Host.disk.utilization	1m	it alar...	>	90	%
常用基础模板_memory_usedu	Host.mem.usedutilization I...	1m	it alar...	>	90	%
常用基础模板_InternetOutRat	Internet Outbound Bandwi...	1m	it alar...	>	90	%
常用基础模板_agent_heartbe	Heartbeat not Detected	1m				

Mute for : 24h

Effective Period : 00:00 To: 23:59

3 **Notification Method**

Notification Contact :

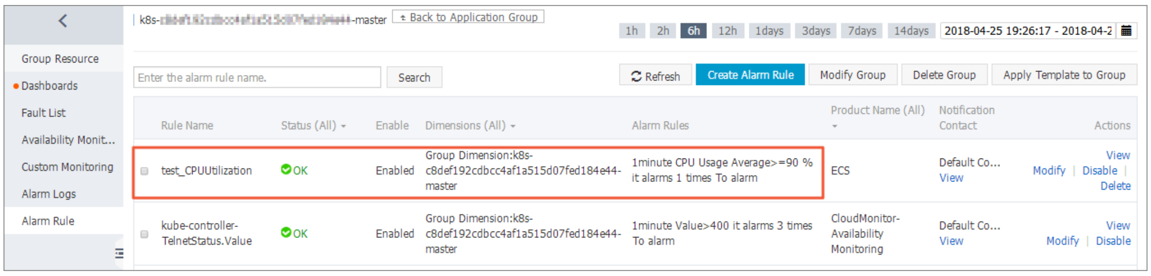
Contact Group All — Search

Quickly create a contact group

Selected Groups 1 count All

云账号报警联系人

3. Click **Confirm**. The created alarm rule is displayed on the **Alarm Rule** page.



k8s-[redacted]-master ↥ Back to Application Group

1h 2h 6h 12h 1days 3days 7days 14days 2018-04-25 19:26:17 - 2018-04-2

Group Resource
Dashboards
Fault List
Availability Monit...
Custom Monitoring
Alarm Logs
Alarm Rule

Enter the alarm rule name. Search — Refresh | Create Alarm Rule | Modify Group | Delete Group | Apply Template to Group

Rule Name	Status (All)	Enable	Dimensions (All)	Alarm Rules	Product Name (All)	Notification Contact	Actions
test_CPUUtilization	OK	Enabled	Group Dimension:k8s-c8def192cdbcc4af1a515d07fed184e44-master	1minute CPU Usage Average>=90 % it alarms 1 times To alarm	ECS	Default Co... View	View Modify | Disable | Delete
kube-controller-TelnetStatus.Value	OK	Enabled	Group Dimension:k8s-c8def192cdbcc4af1a515d07fed184e44-master	1minute Value>400 it alarms 3 times To alarm	CloudMonitor-Availability Monitoring	Default Co... View	View Modify | Disable

What's next

More features are provided to meet your resource monitoring requirements, such as fault list, event monitoring, availability monitoring, and log monitoring. You can find them in the left-side navigation pane.

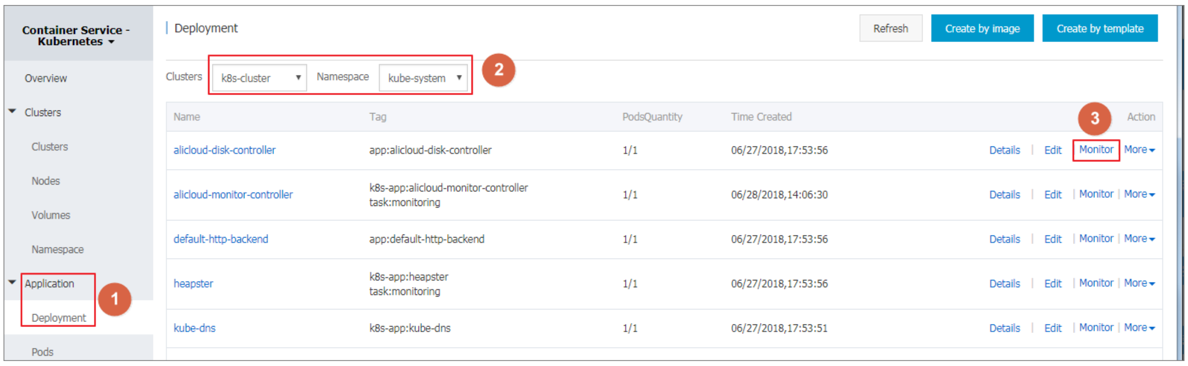
1.11.3 Integration and usage with CloudMonitor

Prerequisites

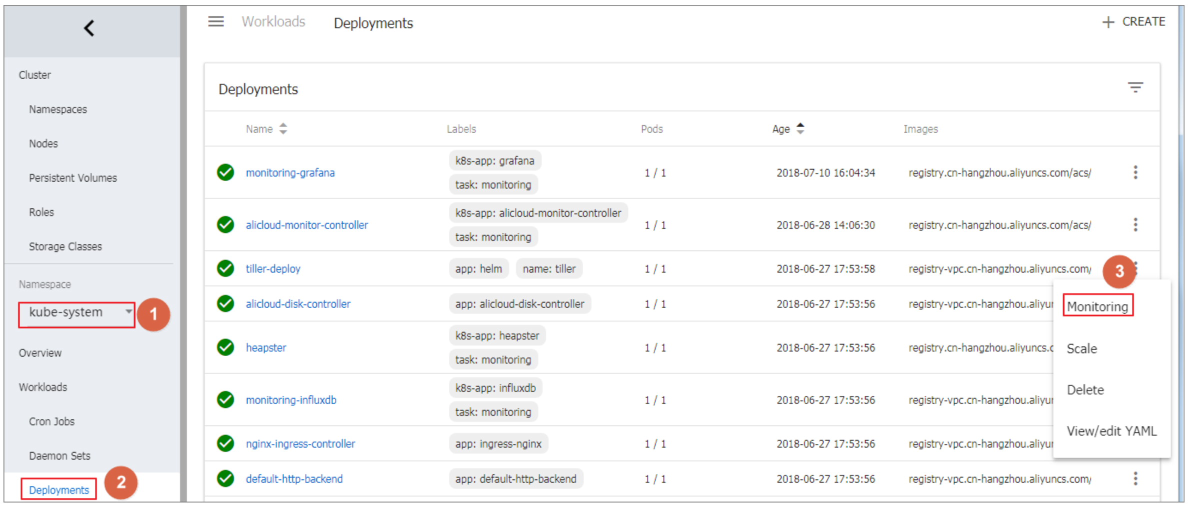
Check whether `alicloud-monitor-controller` has been deployed in the `kube-system` namespace. If not, upgrade the version of the cluster.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Deployment** in the left-side navigation pane.
3. Select the target deployment, click **Monitor** on the right. You can also click **Monitor** on the Deployment page of the built-in kubernetes dashboard.

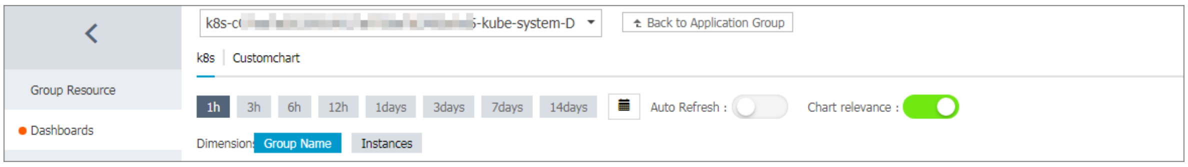


Name	Tag	PodsQuantity	Time Created	Action
alicloud-disk-controller	app:alicloud-disk-controller	1/1	06/27/2018,17:53:56	Details | Edit | Monitor | More
alicloud-monitor-controller	k8s-app:alicloud-monitor-controller task:monitoring	1/1	06/28/2018,14:06:30	Details | Edit | Monitor | More
default-http-backend	app:default-http-backend	1/1	06/27/2018,17:53:56	Details | Edit | Monitor | More
heapster	k8s-app:heapster task:monitoring	1/1	06/27/2018,17:53:56	Details | Edit | Monitor | More
kube-dns	k8s-app:kube-dns	1/1	06/27/2018,17:53:51	Details | Edit | Monitor | More

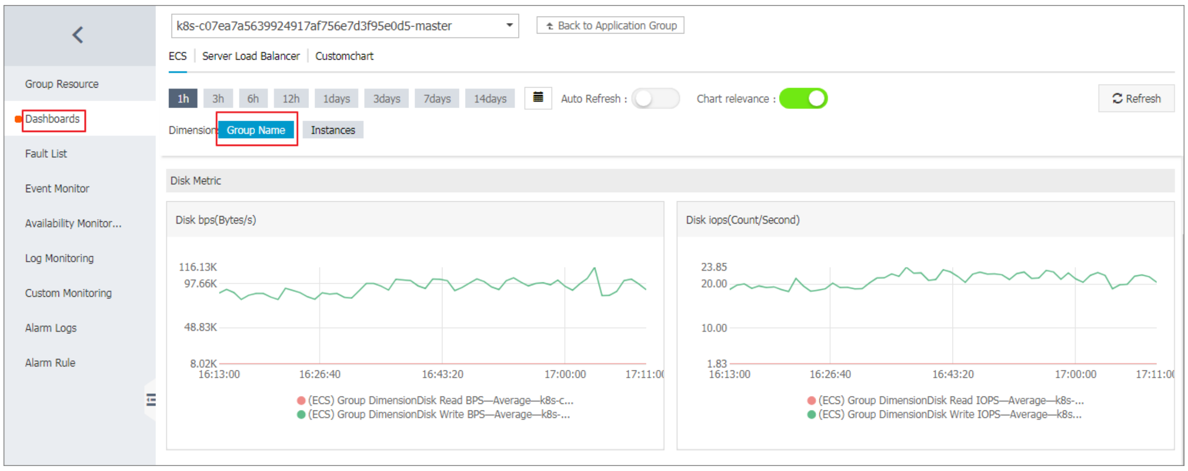


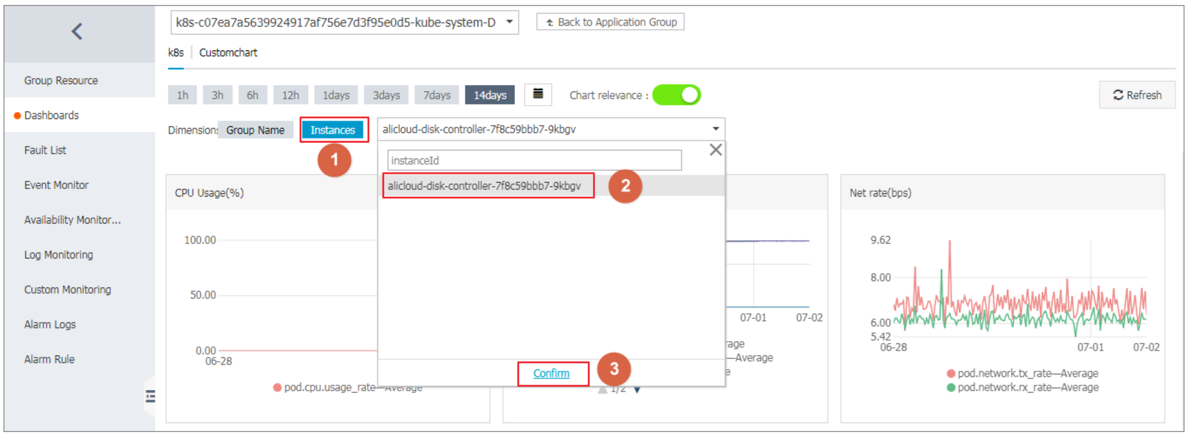
Name	Labels	Pods	Age	Images
monitoring-grafana	k8s-app: grafana task: monitoring	1 / 1	2018-07-10 16:04:34	registry.cn-hangzhou.aliyuncs.com/acs/
alicloud-monitor-controller	k8s-app: alicloud-monitor-controller task: monitoring	1 / 1	2018-06-28 14:06:30	registry.cn-hangzhou.aliyuncs.com/acs/
tiller-deploy	app: helm name: tiller	1 / 1	2018-06-27 17:53:58	registry-vpc.cn-hangzhou.aliyuncs.com/
alicloud-disk-controller	app: alicloud-disk-controller	1 / 1	2018-06-27 17:53:56	registry-vpc.cn-hangzhou.aliyu
heapster	k8s-app: heapster task: monitoring	1 / 1	2018-06-27 17:53:56	registry.cn-hangzhou.aliyuncs.c
monitoring-influxdb	k8s-app: influxdb task: monitoring	1 / 1	2018-06-27 17:53:56	registry-vpc.cn-hangzhou.aliyu
nginx-ingress-controller	app: ingress-nginx	1 / 1	2018-06-27 17:53:56	registry-vpc.cn-hangzhou.aliyu
default-http-backend	app: default-http-backend	1 / 1	2018-06-27 17:53:56	registry-vpc.cn-hangzhou.aliyuncs.com/

In this case, you jump to the corresponding Application group details page of CloudMonitor.

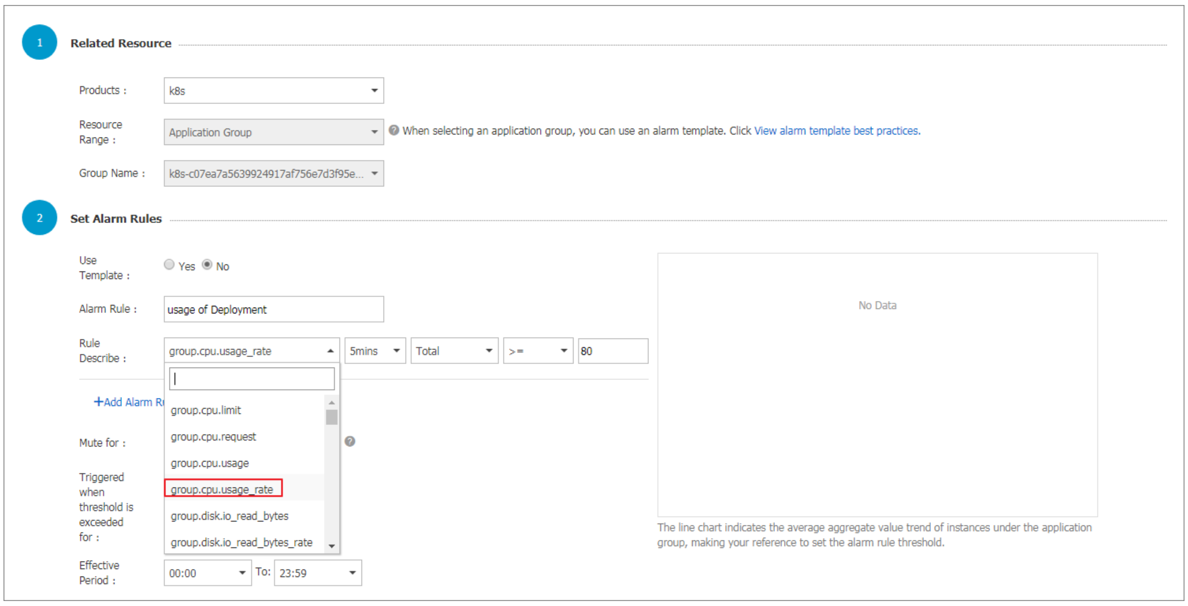


4. Application group supports monitoring in two dimensions: **group** and **instance**.




k8s-c07ea7a5639924917af756e7d3f95e0d5-kube-system-D
Back to Application Group
k8s
Customchart
Chart relevance :
Refresh
Dimension
Group Name
Instances
1
alicloud-disk-controller-7f8c59bbb7-9kbgv
instanceId
alicloud-disk-controller-7f8c59bbb7-9kbgv
2
Confirm
3
CPU Usage(%)
Net rate(bps)
pod.cpu.usage_rate—Average
pod.network.tx_rate—Average
pod.network.rx_rate—Average

5. For alarm settings, the index of group level starts with **`group`**, and the instance level index starts with **pod**.



1 Related Resource

Products : k8s

Resource Range : Application Group

When selecting an application group, you can use an alarm template. Click View alarm template best practices.

Group Name : k8s-c07ea7a5639924917af756e7d3f95e...

2 Set Alarm Rules

Use Template : Yes [✓ No]

Alarm Rule : usage of Deployment

Rule Describe : group.cpu.usage_rate 5mins Total >= 80

Mute for :

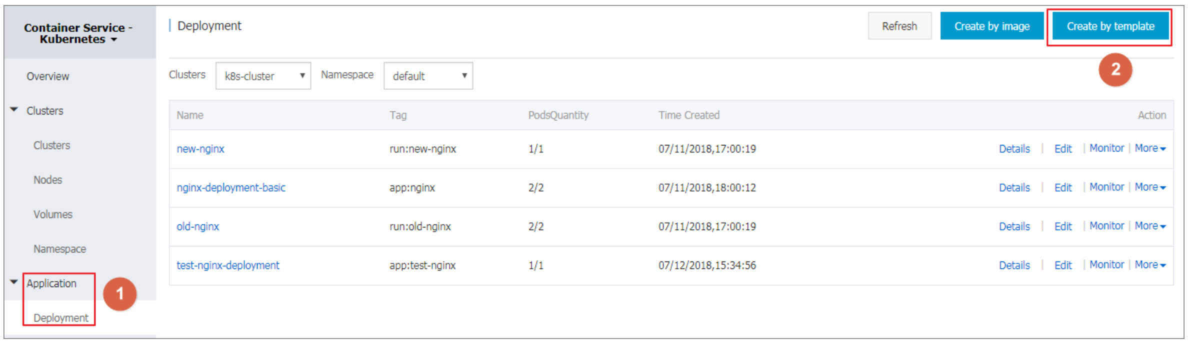
Triggered when threshold is exceeded for :

Effective Period : 00:00 To: 23:59

The line chart indicates the average aggregate value trend of instances under the application group, making your reference to set the alarm rule threshold.

Upgrade cluster version

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane to enter the Deployment List page. Click **Create by template** in the upper-right corner.

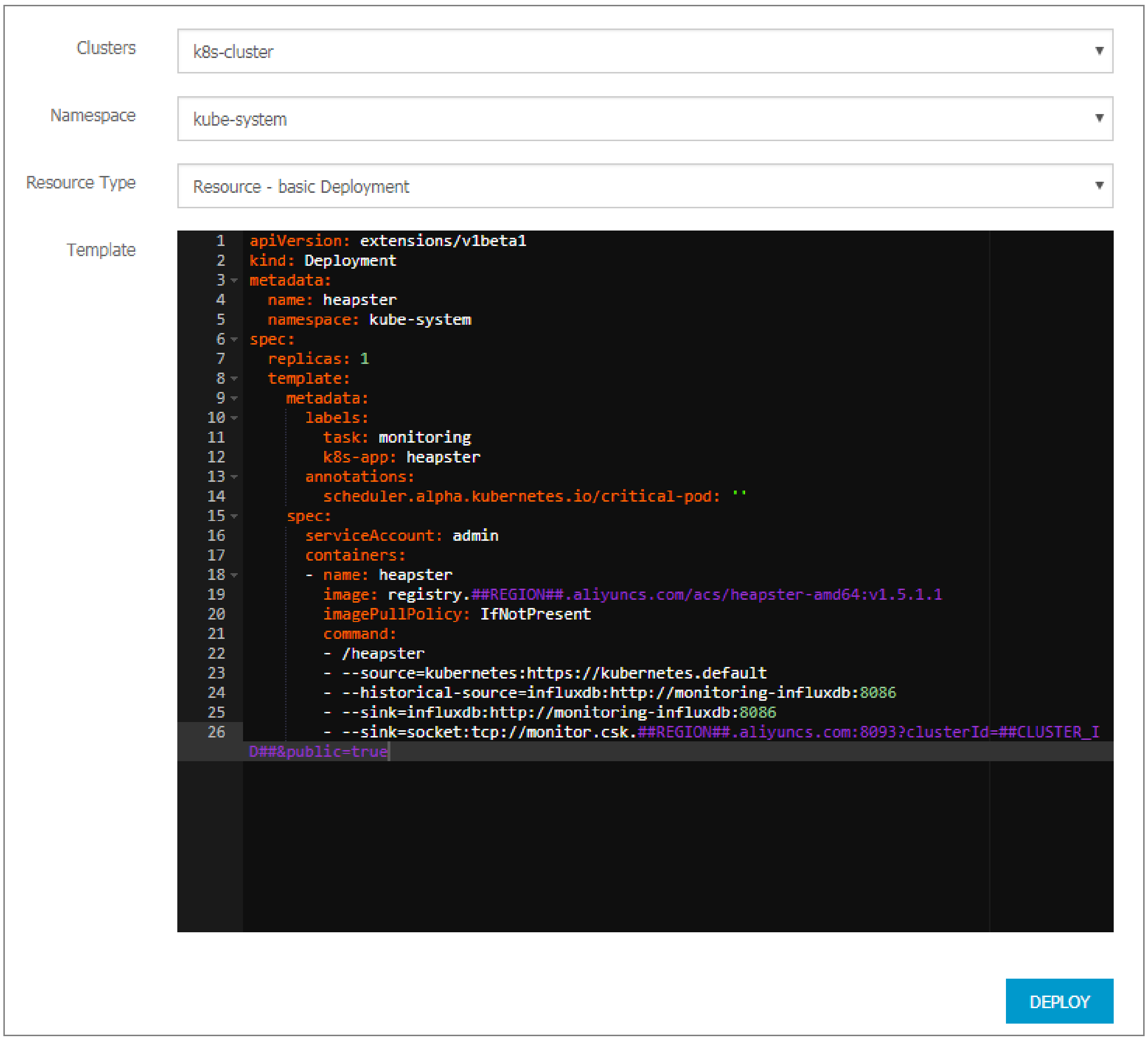


Name	Tag	PodsQuantity	Time Created	Action
new-nginx	run:new-nginx	1/1	07/11/2018,17:00:19	Details | Edit | Monitor | More
nginx-deployment-basic	app:nginx	2/2	07/11/2018,18:00:12	Details | Edit | Monitor | More
old-nginx	run:old-nginx	2/2	07/11/2018,17:00:19	Details | Edit | Monitor | More
test-nginx-deployment	app:test-nginx	1/1	07/12/2018,15:34:56	Details | Edit | Monitor | More

3. Select the target cluster, kube-system namespace, and use the following sample template. Then click **Create**.

Note:

Replace **`REGION`** and **`CLUSTER_ID`** with your actual cluster information, and redeploy heapster yaml template.



Clusters: k8s-cluster

Namespace: kube-system

Resource Type: Resource - basic Deployment

Template:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: heapster
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        task: monitoring
        k8s-app: heapster
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
    spec:
      serviceAccount: admin
      containers:
      - name: heapster
        image: registry.##REGION##.aliyuncs.com/acs/heapster-amd64:v1.5.1.1
        imagePullPolicy: IfNotPresent
        command:
        - /heapster
        - --source=kubernetes:https://kubernetes.default
        - --historical-source=influxdb:http://monitoring-influxdb:8086
        - --sink=influxdb:http://monitoring-influxdb:8086
        - --sink=socket:tcp://monitor.csk.##REGION##.aliyuncs.com:8093?clusterId=##CLUSTER_I
D##&public=true
```

An example of heapster template is as follows. If you have an earlier version of the heapster in the cluster, you can log on to the Kubernetes cluster and run the `kubectl apply -f xxx.yaml` command to upgrade it.

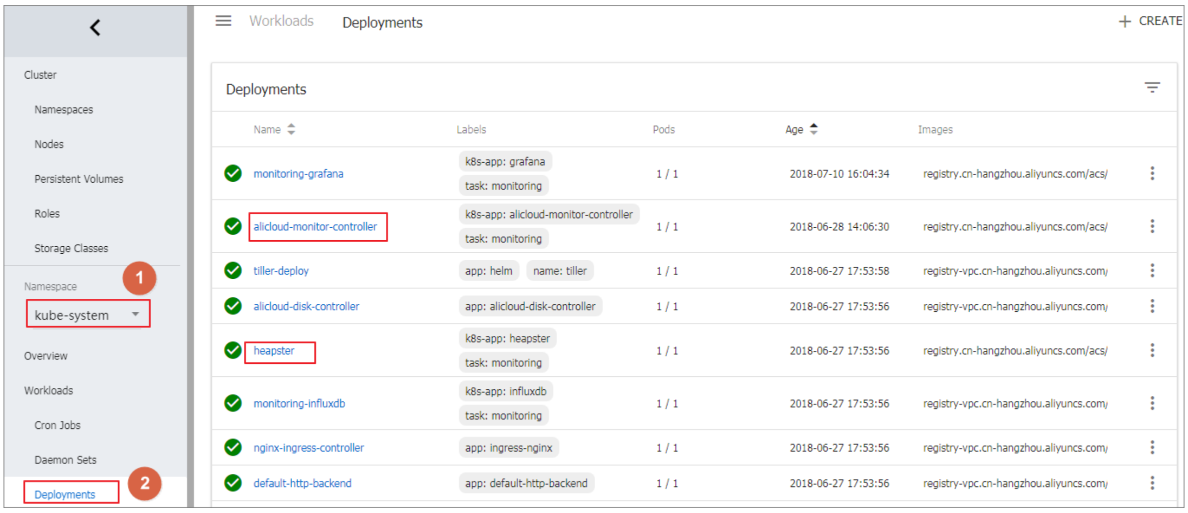
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: heapster
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        task: monitoring
        k8s-app: heapster
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
    spec:
      serviceAccount: admin
      containers:
      - name: heapster
```

```
        image: registry. ##REGION##.aliyuncs.com/acs/heapster-amd64:
v1.5.1.1
        imagePullPolicy: IfNotPresent
        command:
        - /heapster
        - --source=kubernetes:https://kubernetes.default
        - --historical-source=influxdb:http://monitoring-influxdb:
8086
        - --sink=influxdb:http://monitoring-influxdb:8086
        - --sink=socket:tcp://monitor.csk. ##REGION##.aliyuncs.com:
8093? clusterId=##CLUSTER_ID##&public=true
```

The example layout of alicloud-monitor-controller is as follows. Run the `kubectl create -f xxx.yaml` command to deploy alicloud-monitor-controller.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: alicloud-monitor-controller
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        task: monitoring
        k8s-app: alicloud-monitor-controller
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
    spec:
      hostNetwork: true
      tolerations:
      - effect: NoSchedule
        operator: Exists
        key: node-role.kubernetes.io/master
      - effect: NoSchedule
        operator: Exists
        key: node.cloudprovider.kubernetes.io/uninitialized
      serviceAccount: admin
      containers:
      - name: alicloud-monitor-controller
        image: registry. ##REGION##.aliyuncs.com/acs/alicloud-
monitor-controller:v1.0.0
        imagePullPolicy: IfNotPresent
        command:
        - /alicloud-monitor-controller
        - agent
        - --regionId=##REGION##
        - --clusterId=##CLUSTER_ID##
        - --logtostderr
        - --v=4
```

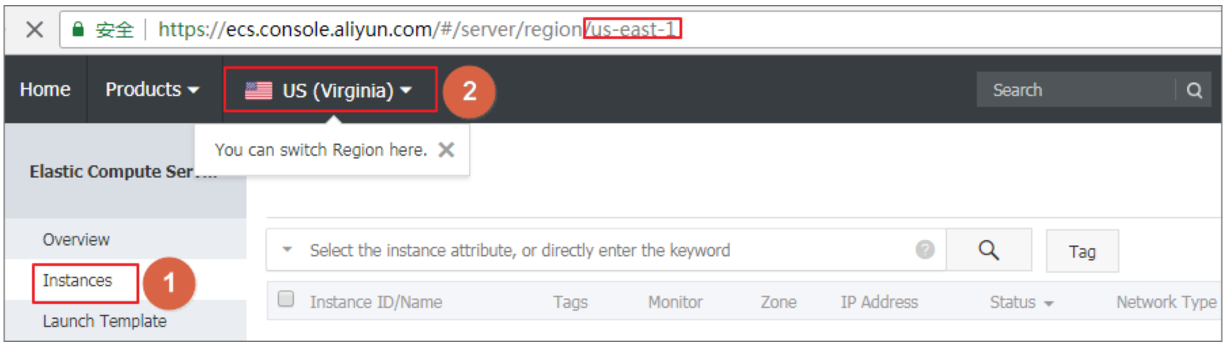
4. Go to the Kubernetes console. In the kube-system namespace, you can see that the two deployments are running, and the upgrade is complete.



Deployments

Name	Labels	Pods	Age	Images
monitoring-grafana	k8s-app: grafana task: monitoring	1 / 1	2018-07-10 16:04:34	registry.cn-hangzhou.aliyuncs.com/acs/
alicloud-monitor-controller	k8s-app: alicloud-monitor-controller task: monitoring	1 / 1	2018-06-28 14:06:30	registry.cn-hangzhou.aliyuncs.com/acs/
tiller-deploy	app: helm name: tiller	1 / 1	2018-06-27 17:53:58	registry-vpc.cn-hangzhou.aliyuncs.com/
alicloud-disk-controller	app: alicloud-disk-controller	1 / 1	2018-06-27 17:53:56	registry-vpc.cn-hangzhou.aliyuncs.com/
heapster	k8s-app: heapster task: monitoring	1 / 1	2018-06-27 17:53:56	registry.cn-hangzhou.aliyuncs.com/acs/
monitoring-influxdb	k8s-app: influxdb task: monitoring	1 / 1	2018-06-27 17:53:56	registry-vpc.cn-hangzhou.aliyuncs.com/
nginx-ingress-controller	app: ingress-nginx	1 / 1	2018-06-27 17:53:56	registry-vpc.cn-hangzhou.aliyuncs.com/
default-http-backend	app: default-http-backend	1 / 1	2018-06-27 17:53:56	registry-vpc.cn-hangzhou.aliyuncs.com/

If you do not know the REGION information, you can go to the ECS console and select the region where your cluster resides. The last segment of the page URL address is REGION.



1.11.4 Use Grafana to display monitoring data

Prerequisites

- You have successfully created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- In this example, use the Grafana with built-in monitoring templates and the image address is `registry.cn-hangzhou.aliyuncs.com/acs/grafana:5.0.4`.

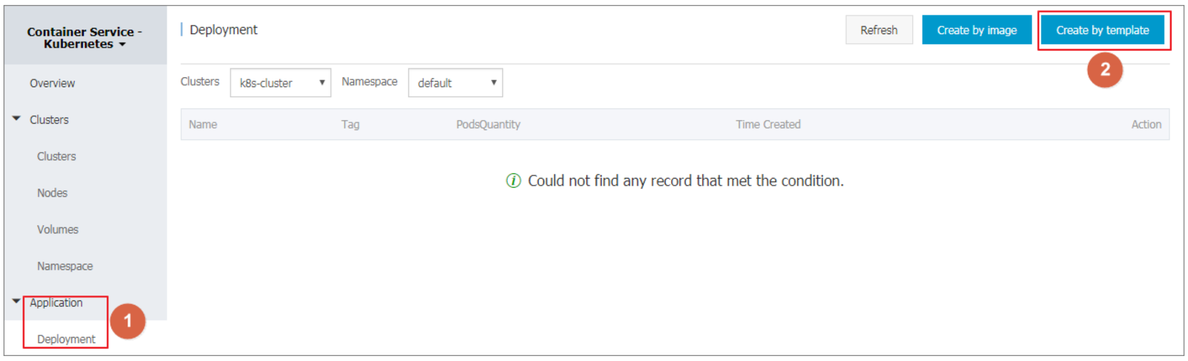
Context

Among Kubernetes monitoring solutions, compared with open-source solutions such as Prometheus, the combination of Heapster + InfluxDB + Grafana is more simple and direct. Heapster not only collects monitoring data in Kubernetes, but also is relied on by the monitoring interface of the console and the POD auto scaling of HPA. Therefore, Heapster is an essential component of Kubernetes. An Alibaba Cloud Kubernetes cluster has the built-in Heapster +

InfluxDB combination. To display the monitoring data, you must configure an available Grafana and the corresponding dashboard.

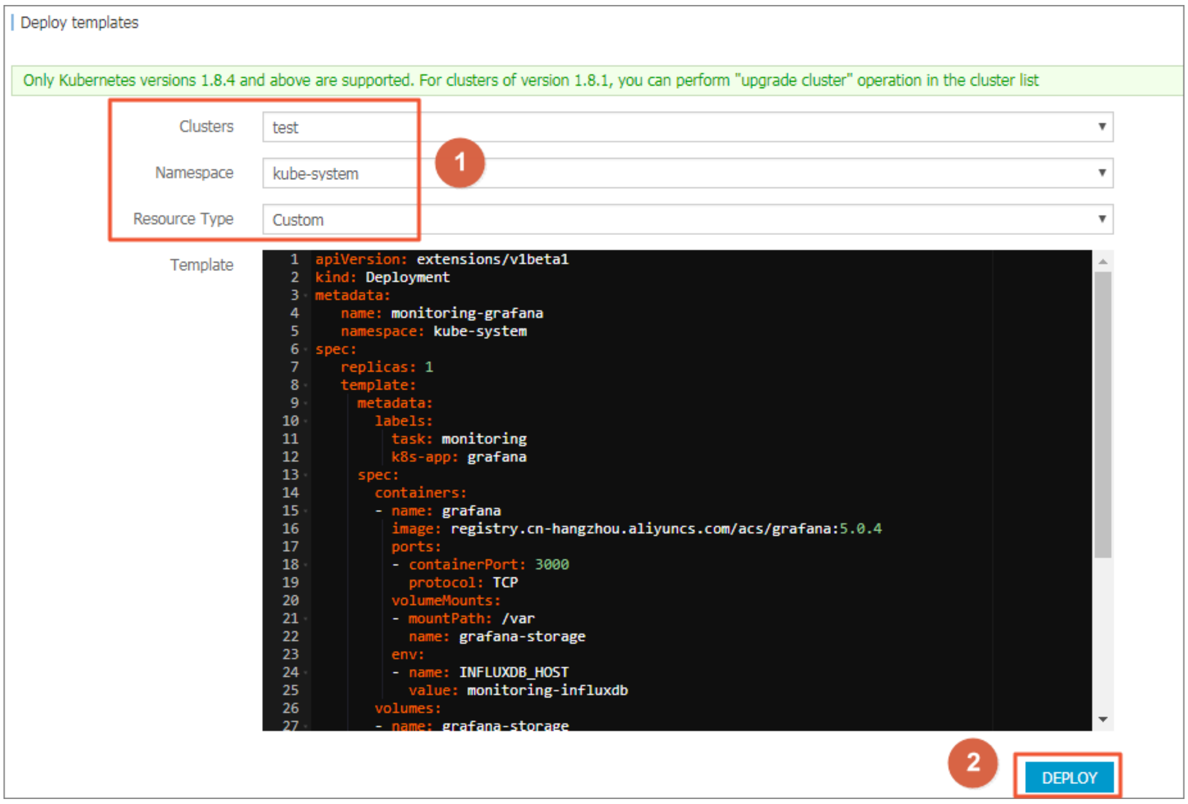
Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane.
3. Click **Create by template** in the upper-right corner.



Could not find any record that met the condition.

4. Configure the template to create the deployment and service of Grafana. After completing the configurations, click **DEPLOY**.
   - Clusters: Select a cluster.
   - Namespace: Select the namespace to which the resource object belongs, which must be **`kube-system`**.
   - Resource Type: Select Custom in this example. The template must contain a deployment and a service.


Deploy templates
Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list
Clusters
test
Namespace
kube-system
Resource Type
Custom
Template
1
2
DEPLOY

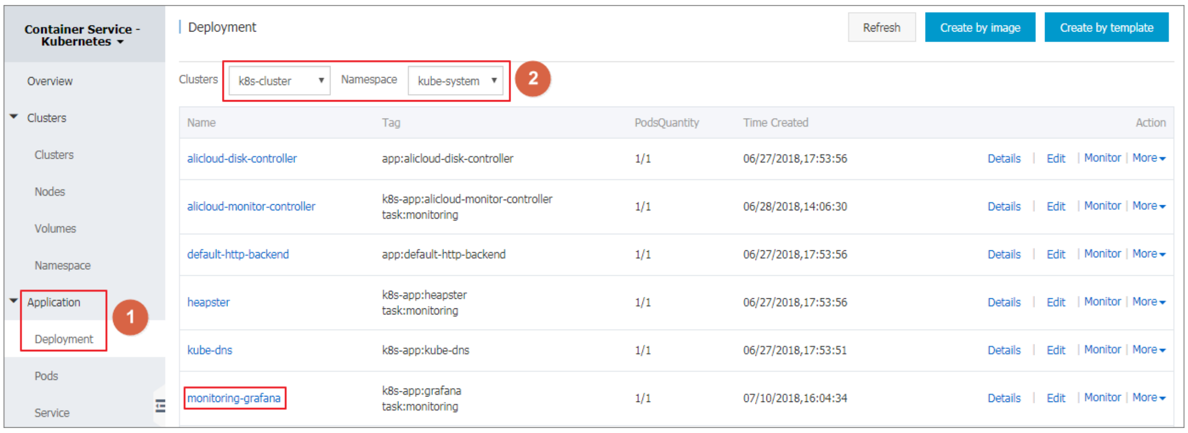
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
   name: monitoring-grafana
   namespace: kube-system
spec:
   replicas: 1
   template:
     metadata:
       labels:
         task: monitoring
         k8s-app: grafana
     spec:
       containers:
       - name: grafana
         image: registry.cn-hangzhou.aliyuncs.com/acs/grafana:5.0.4
         ports:
         - containerPort: 3000
           protocol: TCP
         volumeMounts:
         - mountPath: /var
           name: grafana-storage
         env:
         - name: INFLUXDB_HOST
           value: monitoring-influxdb
       volumes:
       - name: grafana-storage
         emptyDir: {}
```

The orchestration template in this example is as follows:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
   name: monitoring-grafana
   namespace: kube-system
spec:
   replicas: 1
   template:
     metadata:
       labels:
         task: monitoring
         k8s-app: grafana
     spec:
       containers:
       - name: grafana
         image: registry.cn-hangzhou.aliyuncs.com/acs/grafana:5.0.4
         ports:
         - containerPort: 3000
           protocol: TCP
         volumeMounts:
         - mountPath: /var
           name: grafana-storage
         env:
         - name: INFLUXDB_HOST
           value: monitoring-influxdb
       volumes:
       - name: grafana-storage
         emptyDir: {}
 ---
apiVersion: v1
kind: Service
```

```
metadata:
   name: monitoring-grafana
   namespace: kube-system
spec:
   ports:
   - port: 80
     targetPort: 3000
   type: LoadBalancer
   selector:
     k8s-app: grafana
```

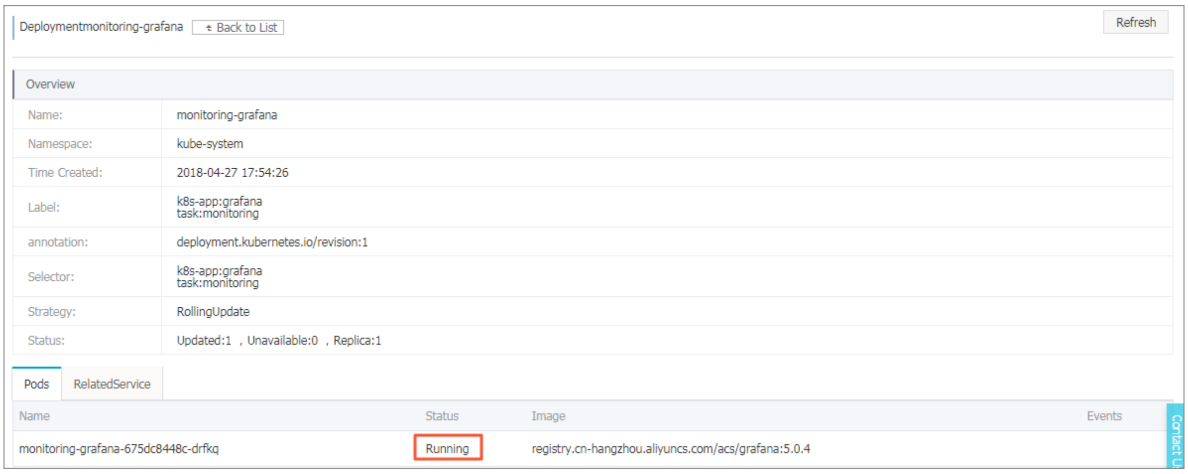
5. Go back to the Deployment page after the successful deployment. Select the cluster from the Clusters drop-down list and then select kube-system from the Namespace drop-down list to view the deployed applications.



Clusters k8s-cluster Namespace kube-system

Name	Tag	PodsQuantity	Time Created	Action
alicloud-disk-controller	app:alicloud-disk-controller	1/1	06/27/2018,17:53:56	Details | Edit | Monitor | More
alicloud-monitor-controller	k8s-app:alicloud-monitor-controller task:monitoring	1/1	06/28/2018,14:06:30	Details | Edit | Monitor | More
default-http-backend	app:default-http-backend	1/1	06/27/2018,17:53:56	Details | Edit | Monitor | More
heapster	k8s-app:heapster task:monitoring	1/1	06/27/2018,17:53:56	Details | Edit | Monitor | More
kube-dns	k8s-app:kube-dns	1/1	06/27/2018,17:53:51	Details | Edit | Monitor | More
monitoring-grafana	k8s-app:grafana task:monitoring	1/1	07/10/2018,16:04:34	Details | Edit | Monitor | More

6. Click the name monitoring-grafana to view the deployment status. Wait until the running status changes to Running.

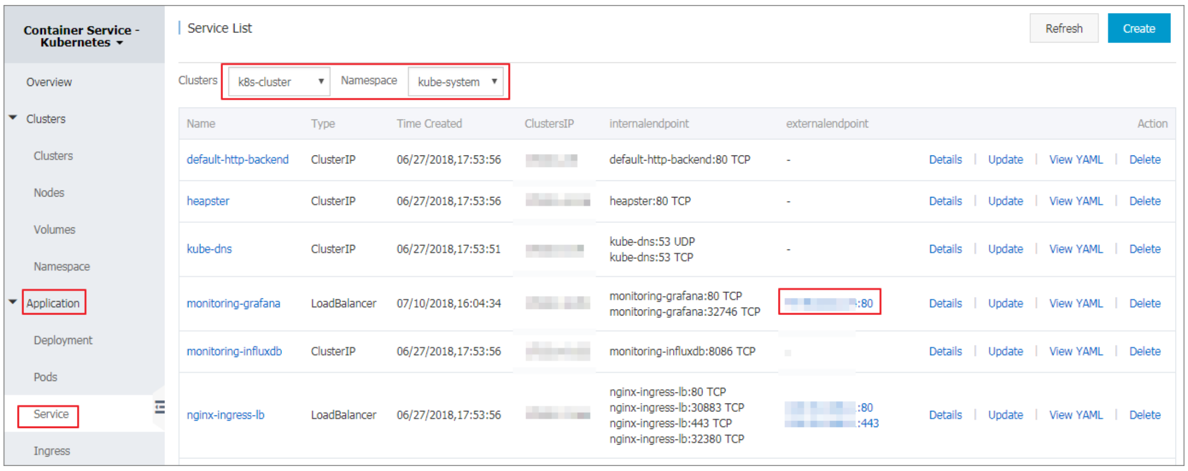


Name:	monitoring-grafana
Namespace:	kube-system
Time Created:	2018-04-27 17:54:26
Label:	k8s-app:grafana task:monitoring
annotation:	deployment.kubernetes.io/revision:1
Selector:	k8s-app:grafana task:monitoring
Strategy:	RollingUpdate
Status:	Updated:1 , Unavailable:0 , Replica:1

Name	Status	Image	Events
monitoring-grafana-675dc8448c-drfkq	Running	registry.cn-hangzhou.aliyuncs.com/acs/grafana:5.0.4	

7. Click **Application** > **Service** in the left-side navigation pane. Select the cluster from the Clusters drop-down list and kube-system from the Namespace drop-down list to view the external endpoint.

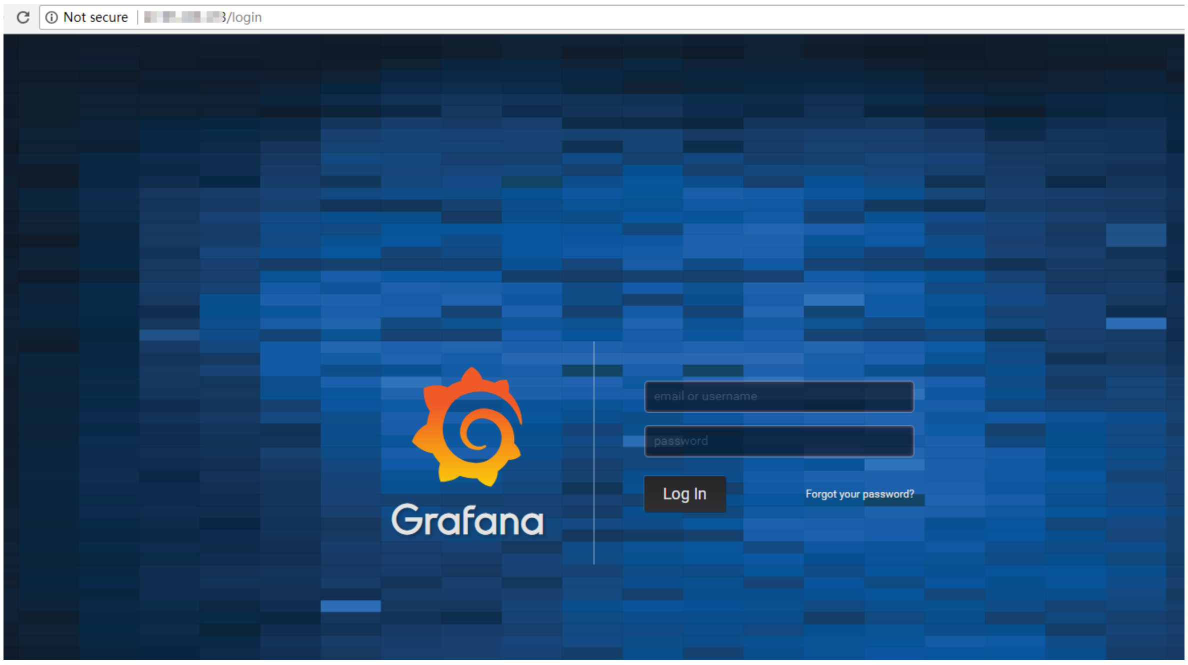
The external endpoint is automatically created by using the LoadBalancer type service. For developers who require more secure access policies, we recommend that you increase the security by adding the external endpoint to the IP whitelist or configuring the certificate.



Name	Type	Time Created	ClustersIP	internalendpoint	externalendpoint	Action
default-http-backend	ClusterIP	06/27/2018,17:53:56	[redacted]	default-http-backend:80 TCP	-	Details | Update | View YAML | Delete
heapster	ClusterIP	06/27/2018,17:53:56	[redacted]	heapster:80 TCP	-	Details | Update | View YAML | Delete
kube-dns	ClusterIP	06/27/2018,17:53:51	[redacted]	kube-dns:53 UDP kube-dns:53 TCP	-	Details | Update | View YAML | Delete
monitoring-grafana	LoadBalancer	07/10/2018,16:04:34	[redacted]	monitoring-grafana:80 TCP monitoring-grafana:32746 TCP	[redacted]:80	Details | Update | View YAML | Delete
monitoring-influxdb	ClusterIP	06/27/2018,17:53:56	[redacted]	monitoring-influxdb:8086 TCP		Details | Update | View YAML | Delete
nginx-ingress-lb	LoadBalancer	06/27/2018,17:53:56	[redacted]	nginx-ingress-lb:80 TCP nginx-ingress-lb:30883 TCP nginx-ingress-lb:443 TCP nginx-ingress-lb:32380 TCP	[redacted]:80 [redacted]:443	Details | Update | View YAML | Delete

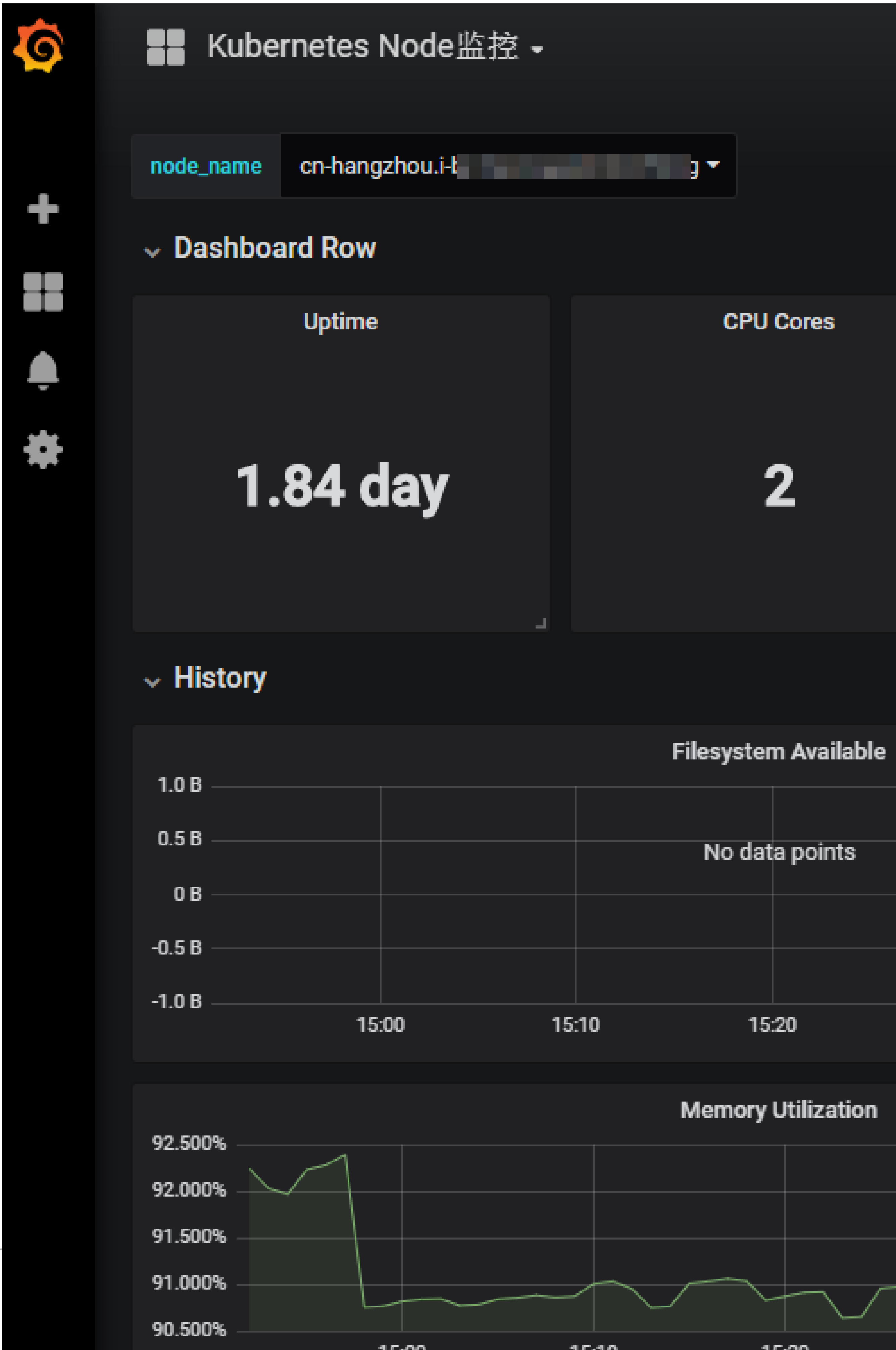
8. Click the **external endpoint** at the right of the monitoring-grafana service to log on to the Grafana monitoring page.

   By default, the username and password of Grafana are both admin. We recommend that you change the password after the logon.



9. Select the built-in monitoring templates to view the monitoring dashboards of the pod and node.

   In this example, the Grafana has two built-in templates, one for displaying physical resources at the node level, and one for displaying resources related to the pod. Developers can also perform more complex presentations by adding custom dashboards or configure resource alarms based on Grafana.



Kubernetes Node监控

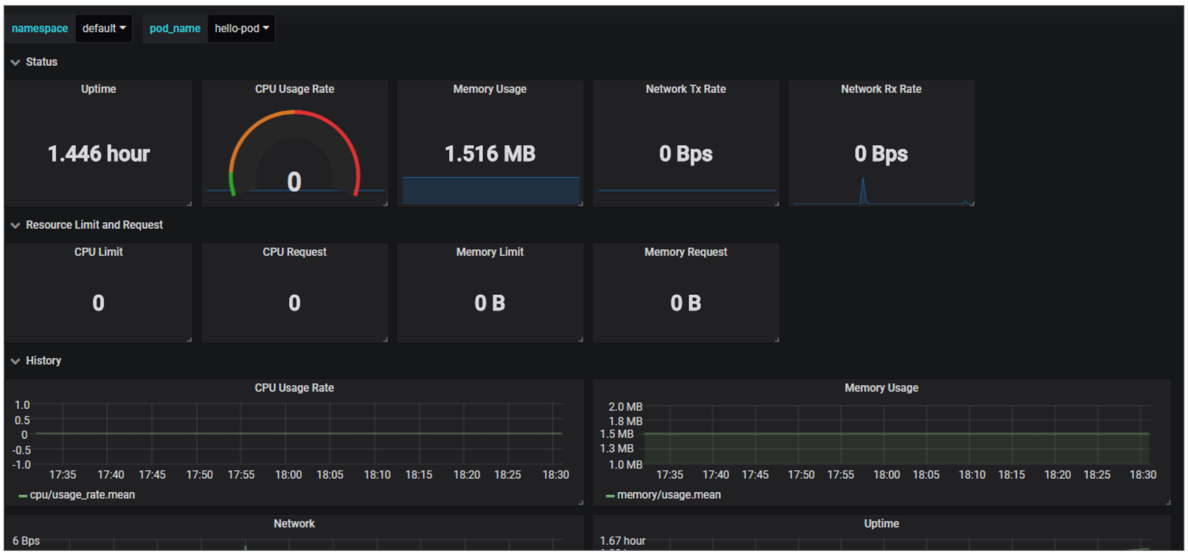
node_name cn-hangzhou.i-[redacted]g

Dashboard Row

Uptime 1.84 day

CPU Cores 2

History



1.11.5 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).

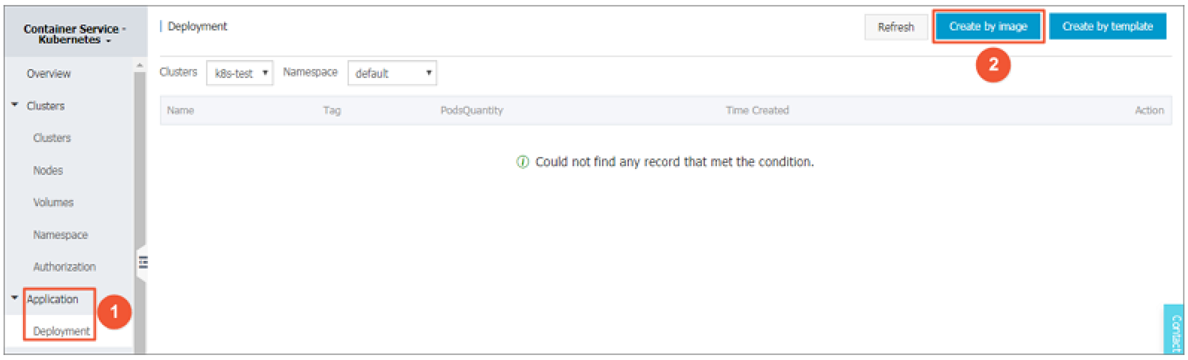
Prerequisites

- 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

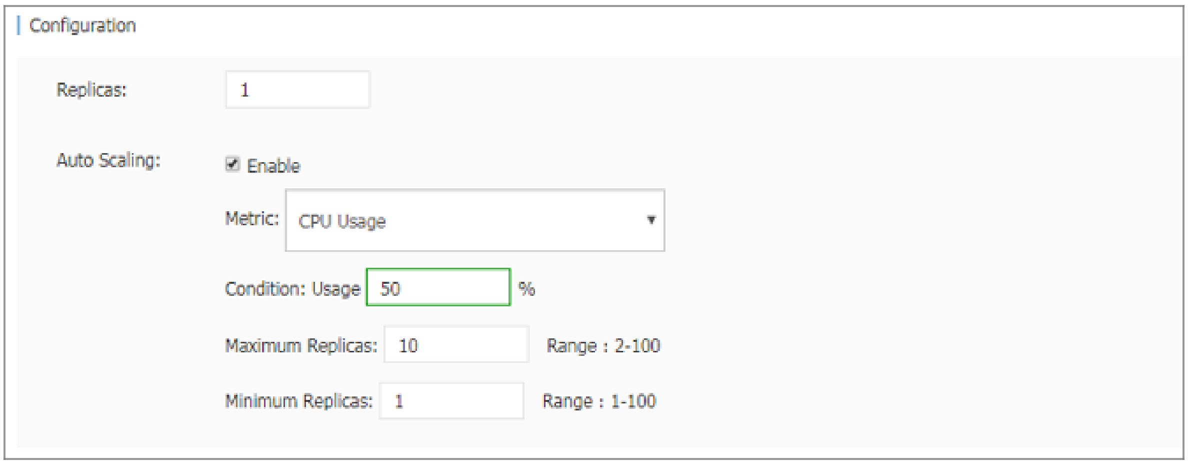
In Alibaba Cloud Container Service, HPA has been integrated. You can easily create it through the Container Service console.

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane. Click **Create by image** in the upper-right corner.



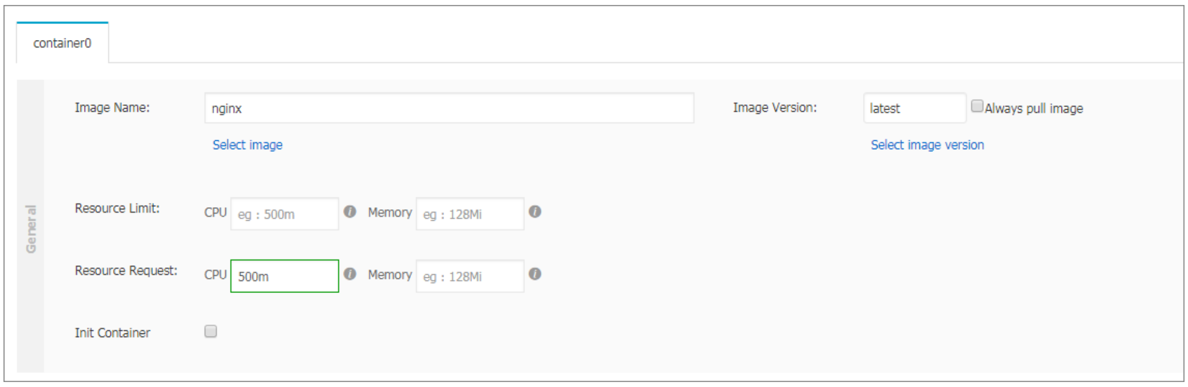
Could not find any record that met the condition.

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.


Configuration
Replicas: 1
Auto Scaling: ☑ Enable
Metric: CPU Usage
Condition: Usage 50 %
Maximum Replicas: 10 Range : 2-100
Minimum Replicas: 1 Range : 1-100

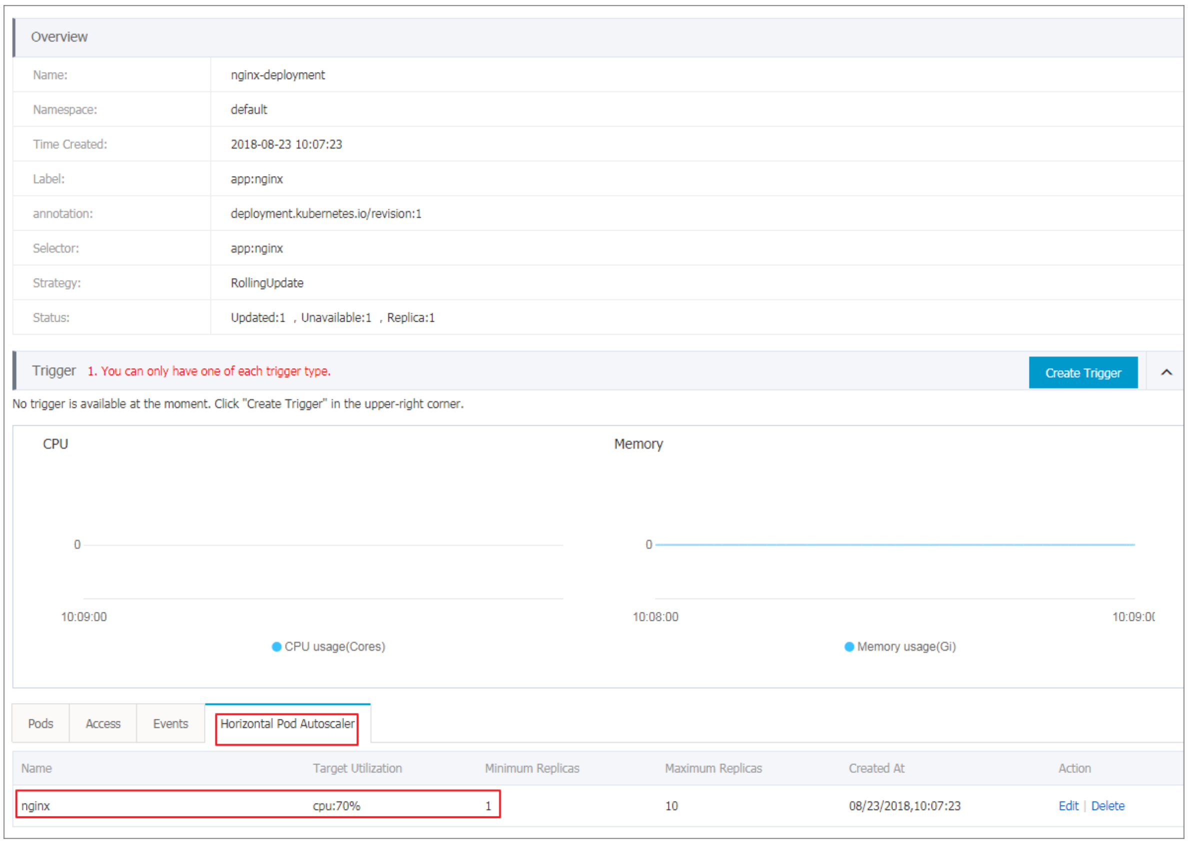
5. Configure the container. Select an image and configure the required resources. Click **Next**.

Note:
You must configure the required resources for the deployment. Otherwise, container auto scaling cannot be achieved.


container0
General
Image Name: nginx
Select image
Image Version: latest ☐ Always pull image
Select image version
Resource Limit: CPU eg : 500m Memory eg : 128Mi
Resource Request: CPU 500m Memory eg : 128Mi
Init Container ☐

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.



Overview

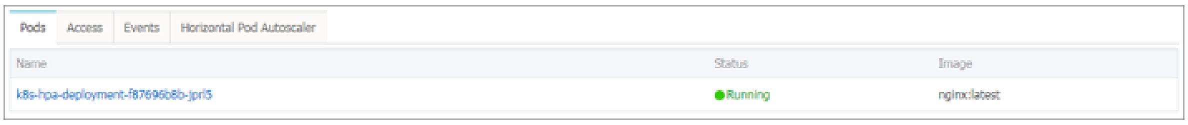
Name:	nginx-deployment
Namespace:	default
Time Created:	2018-08-23 10:07:23
Label:	app:nginx
annotation:	deployment.kubernetes.io/revision:1
Selector:	app:nginx
Strategy:	RollingUpdate
Status:	Updated:1 , Unavailable:1 , Replica:1

Trigger 1. You can only have one of each trigger type.

No trigger is available at the moment. Click "Create Trigger" in the upper-right corner.

Name	Target Utilization	Minimum Replicas	Maximum Replicas	Created At	Action
nginx	cpu:70%	1	10	08/23/2018,10:07:23	Edit | Delete

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.



Name	Status	Image
k8s-hpa-deployment-f87696b8b-jprl5	● Running	nginx:latest

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:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <image_name:
tags>
        ports:
        - containerPort: 80
        resources:
          requests:                         ##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 **`scaleTargetRef`**. In this example, the object is the deployment named nginx.

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
  namespace: default
spec:
  scaleTargetRef:                             ##Bind the HPA to a
deployment named nginx
    apiVersion: apps/v1beta2
    kind: Deployment
    name: nginx
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 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  FailedGetResourceMetric       2m (x6 over 4m)  horizontal-pod
-autoscaler  missing request for cpu on container nginx in pod default
/nginx-deployment-basic-75675f5897-mqzs7
```

```
Warning  FailedComputeMetricsReplicas  2m (x6 over 4m)  horizontal-pod
-autoscaler  failed to get cpu utilization: missing request for cpu on
 container nginx in pod default/nginx-deployment-basic-75675f5
```

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 SuccessfulRescale 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 horizontal ly. When the usage of Nginx pod drops below 50%, the container contracts.

1.11.6 Monitor a Kubernetes cluster and send alarm notificati ons by using DingTalk

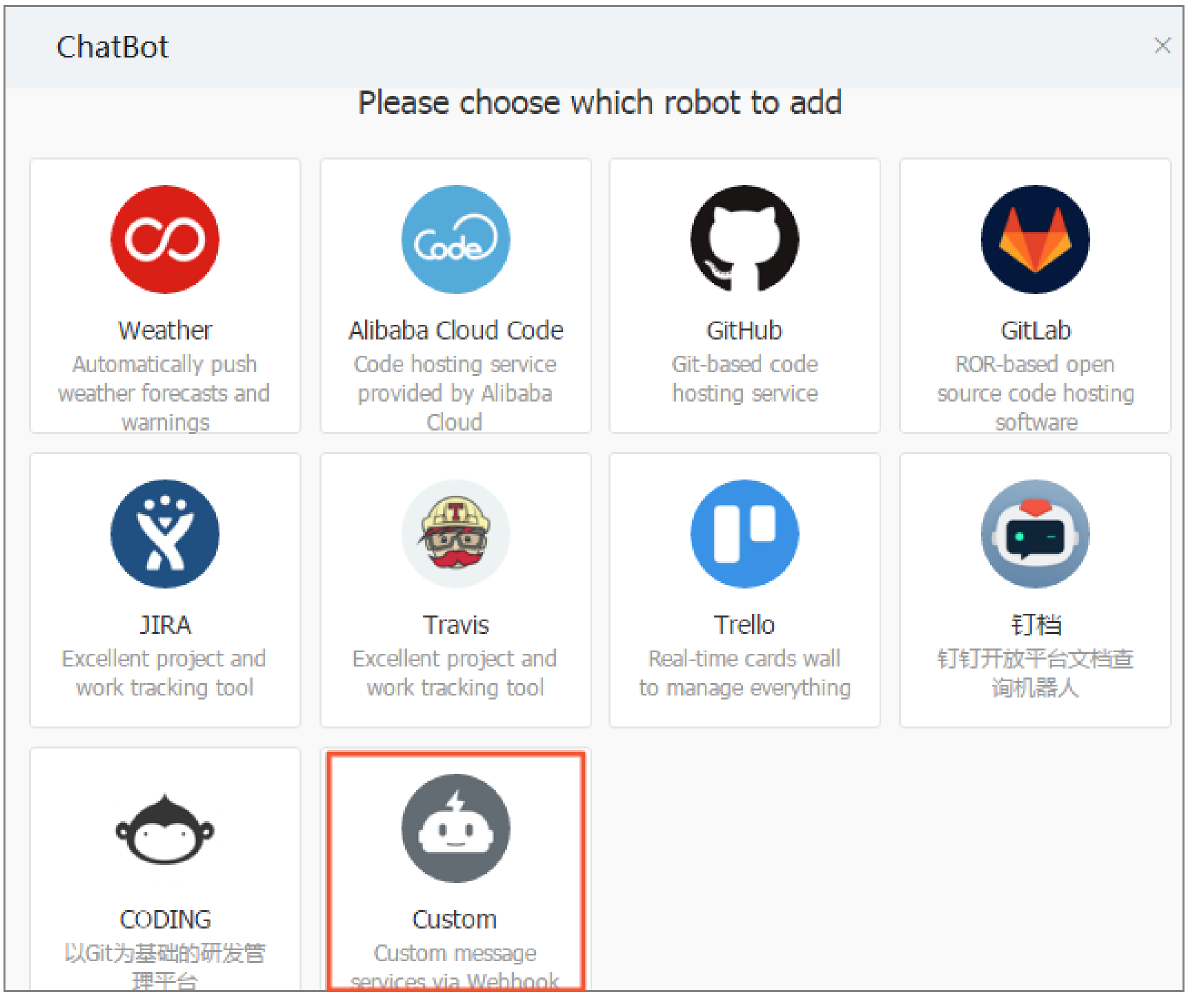
After you deploy a robot in a DingTalk group, the cluster sends a notification of an exception event to the DingTalk group through the robot, implementing real-time monitoring and alarming for cluster exception events.

Context

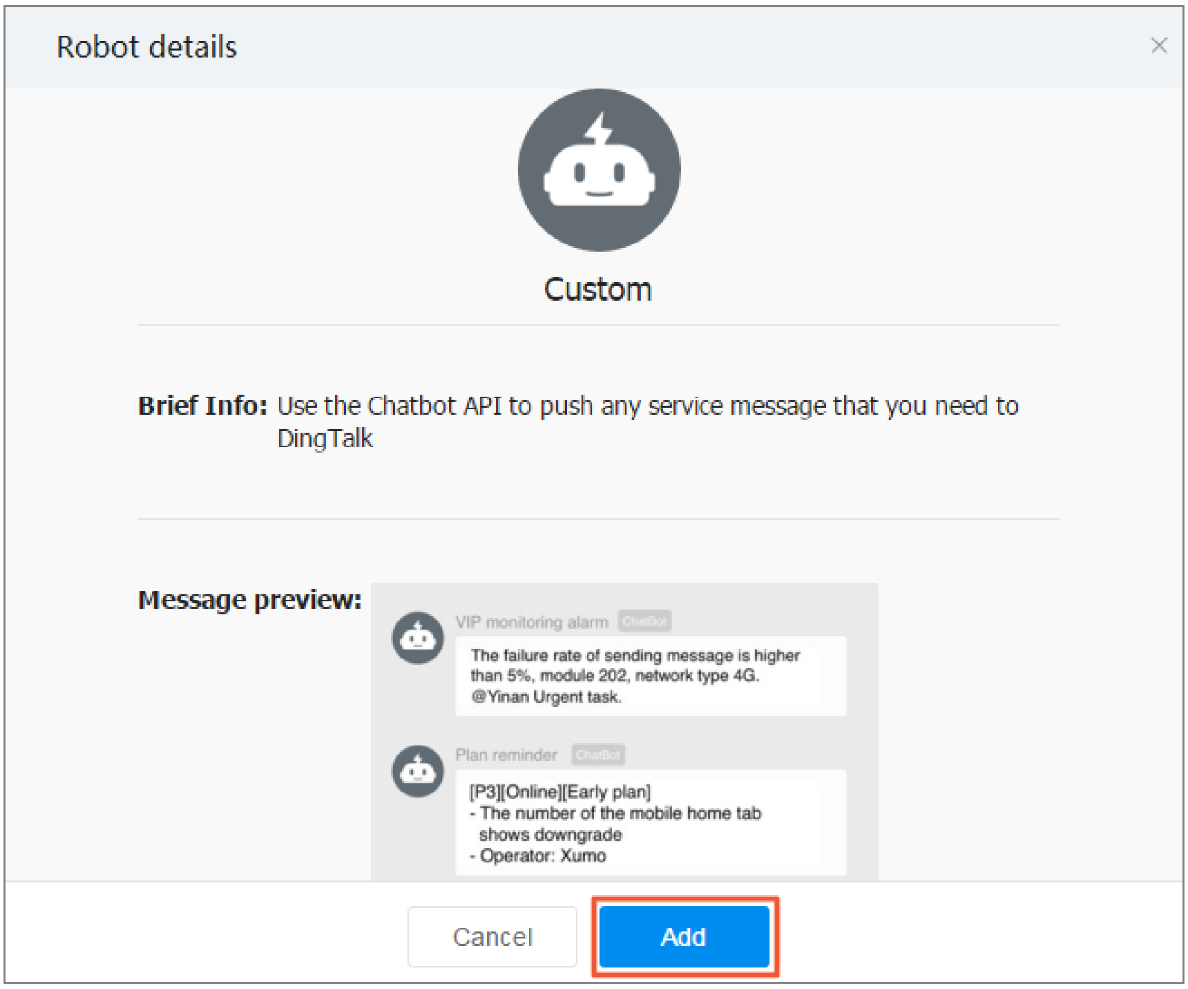
- You have created a DingTalk group .
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

Procedure

1. Click the icon in the upper-right corner of the DingTalk group.
2. Click **ChatBot**. On the **ChatBot** page, select a robot. Select a **Custom** robot.


ChatBot
Please choose which robot to add
Weather
Automatically push weather forecasts and warnings
Alibaba Cloud Code
Code hosting service provided by Alibaba Cloud
GitHub
Git-based code hosting service
GitLab
ROR-based open source code hosting software
JIRA
Excellent project and work tracking tool
Travis
Excellent project and work tracking tool
Trello
Real-time cards wall to manage everything
钉档
钉钉开放平台文档查询机器人
CODING
以Git为基础的研发管理平台
Custom
Custom message services via Webhook

3. On the **Robot details** page, click **Add**.



Brief Info: Use the Chatbot API to push any service message that you need to DingTalk

Message preview:

VIP monitoring alarm ChatBot

The failure rate of sending message is higher than 5%, module 202, network type 4G. @Yinan Urgent task.

Plan reminder ChatBot

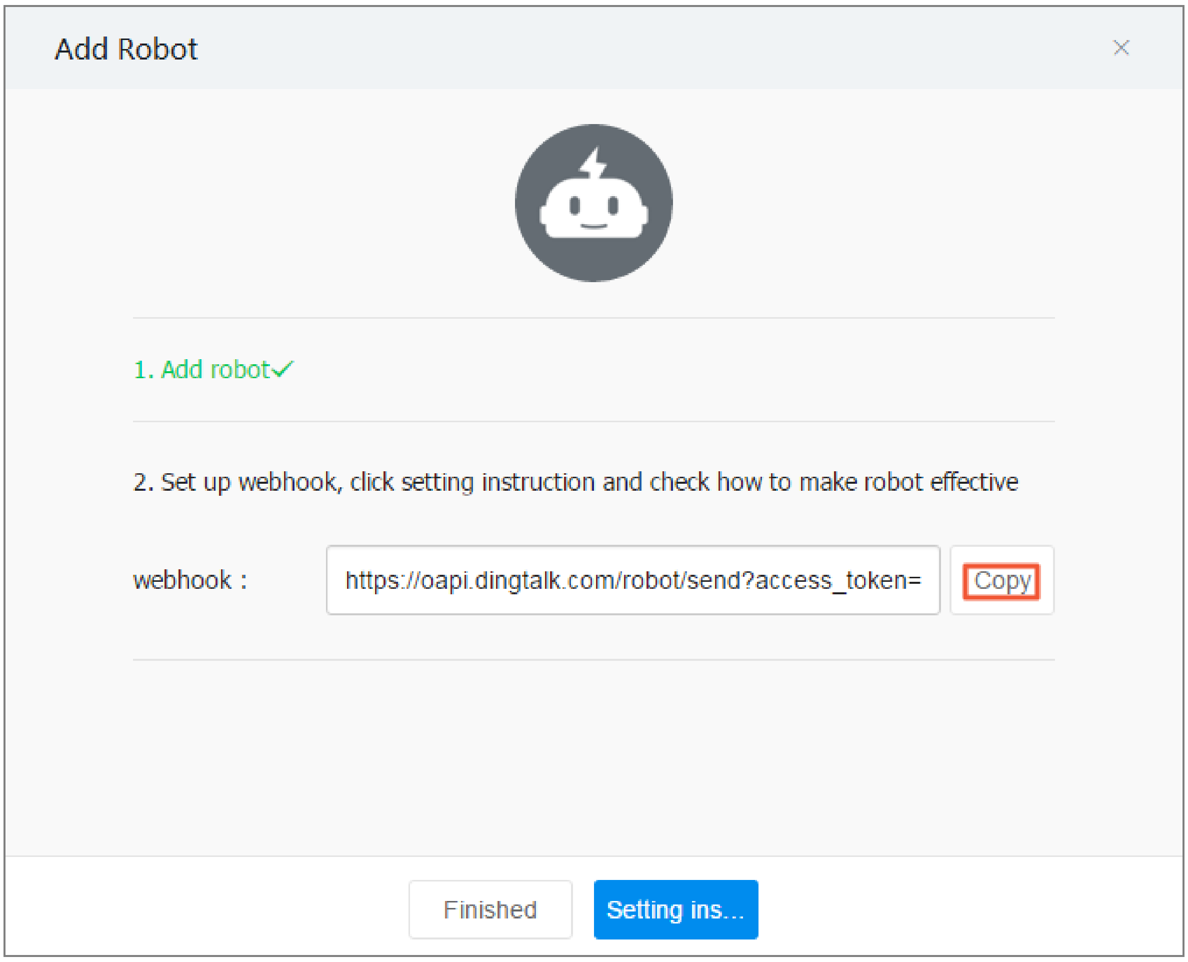
[P3][Online][Early plan]
- The number of the mobile home tab shows downgrade
- Operator: Xumo

4. Configure the following parameters for a robot and then click **Finished**:

Configuration	Description
Edit profile picture	(Optional) Set a profile picture for the robot.
ChatBot Name	The robot name.
Add to Group	The DingTalk group to which the robot is added to.
Enable the outgoing function	(Optional) By perform the @robot operation, you can send messages to a specified external service as well as return response results of the external service to the group. **Note:** We recommend that you do not enable this function.
POST address	The HTTP service address that receives messages. **Note:**

Configuration	Description
	You can configure this parameter after you enable the outgoing function.
Token	The key used to verify that a request is from DingTalk. **Note:** You can configure this parameter after you enable the outgoing function.

5. Click **Copy** to copy the webhook address.



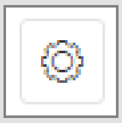
Add Robot

1. Add robot✓

2. Set up webhook, click setting instruction and check how to make robot effective

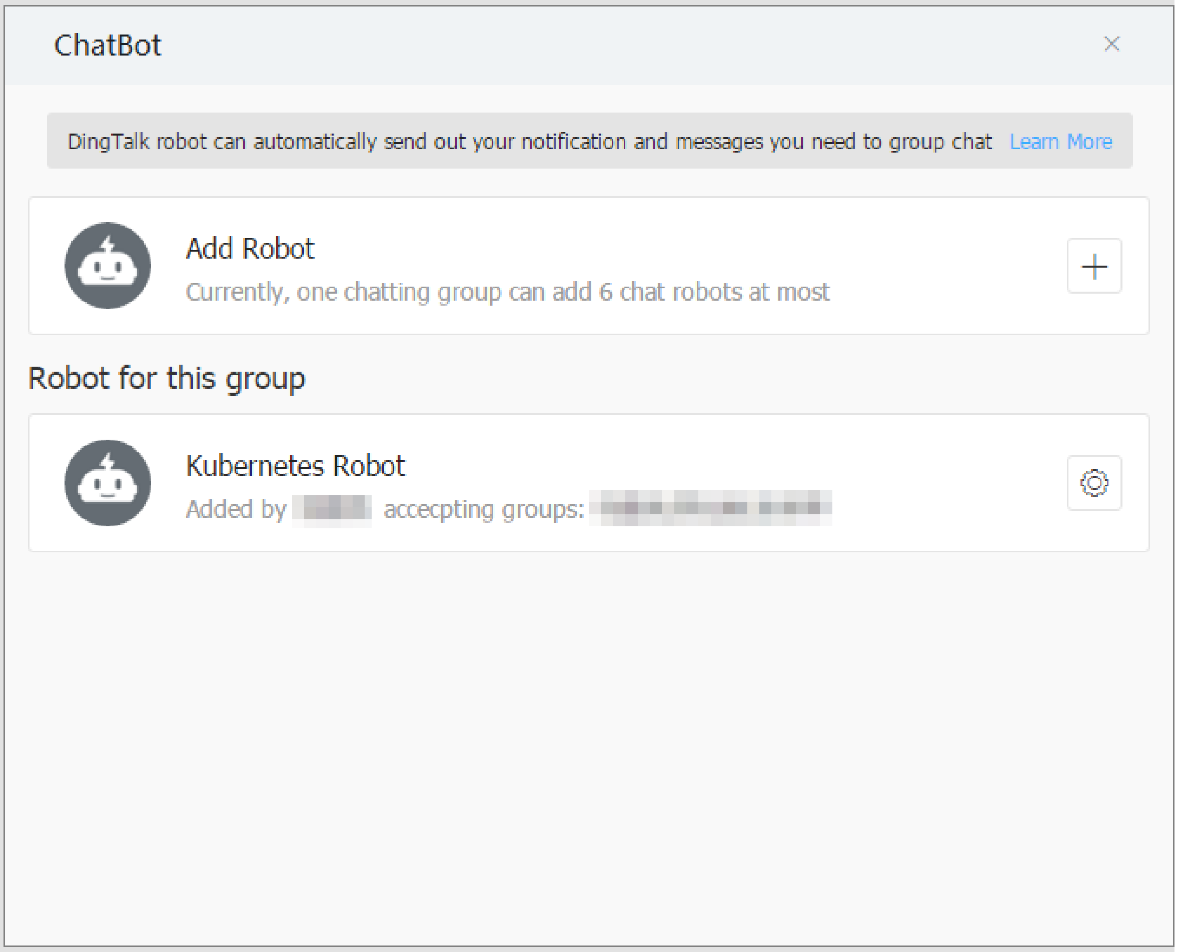
webhook : https://oapi.dingtalk.com/robot/send?access_token=

Note:

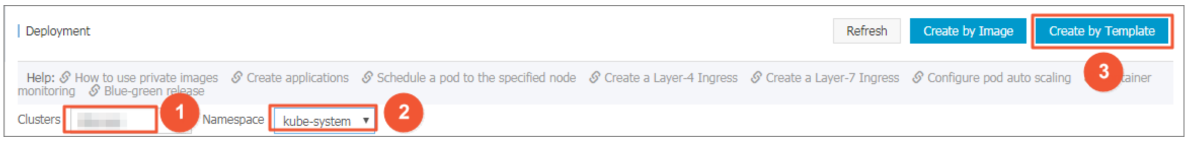
On the **ChatBot** page, click the  icon at the right of a robot and then you can perform following operations:

- Modify the profile picture and name of the robot.

- **Open** or **Close** notifications.
- Reset the webhook address.
- Remove the robot.


ChatBot
DingTalk robot can automatically send out your notification and messages you need to group chat Learn More
Add Robot
Currently, one chatting group can add 6 chat robots at most
Robot for this group
Kubernetes Robot
Added by [redacted] accecpting groups: [redacted]

6. Log on to the *Container Service console*.
7. Under the Kubernetes menu, click **Application** > **Deployment** in the left-side navigation pane.
8. Select a cluster, select the **kube-system** namespace, and click **Create by Template** in the upper-right corner.


Deployment
Refresh
Create by Image
Create by Template
Help: How to use private images Create applications Schedule a pod to the specified node Create a Layer-4 Ingress Create a Layer-7 Ingress Configure pod auto scaling ainer monitoring Blue-green release
Clusters [redacted]
1
Namespace kube-system
2
3

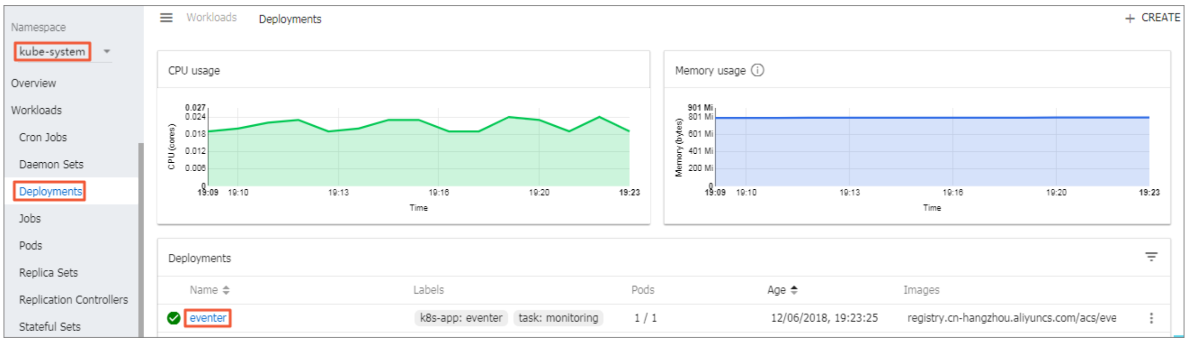
9. Configure a template based on the following parameters, and then click **Deploy**.

Configuration	Description
Clusters	Select a cluster.

Configuration	Description
Namespace	Select a namespace to which resource object belongs. The default namespace is **default**. Select **kube-system**.
Sample template	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. Select **Custom**.

Template	Enter the following custom content:
	`apiVersion: extensions/v1beta1`
	`kind: Deployment`
	`metadata:`
	`  name: eventer`
	`  namespace: kube-system`
	`spec:`
	`  replicas: 1`
	`  template:`
	`    metadata:`
	`      labels:`
	`        task: monitoring`
	`        k8s-app: eventer`
	`      annotations:`
	`        scheduler.alpha.kubernetes.io/critical-pod:`
	` ''`
	`    spec:`
	`      serviceAccount: admin`
	`      containers:`
	`      - name: eventer`
	`        image: registry.cn-hangzhou.aliyuncs.com/`
	`acs/eventer:v1.6.0`
	`        imagePullPolicy: IfNotPresent`
	`        command:`
	`        - /eventer`
	`        - --source=kubernetes:https://kubernetes.`
	`default`
	`        - --sink=dingtalk:[your_webhook_url]&`
	`label=[your_cluster_id]&level       #level cab`
	`be configured as normal or warning. The default`
	`is warning. When the level is configure as normal`
	`, alarm notifications of the normal and warning`
	`levels can be received in the DingTalk group. When`
	` you do not configure the level or configure the`
	`level as warning, alarm notifications of only the`
	`warning level can be received in the DingTalk group`

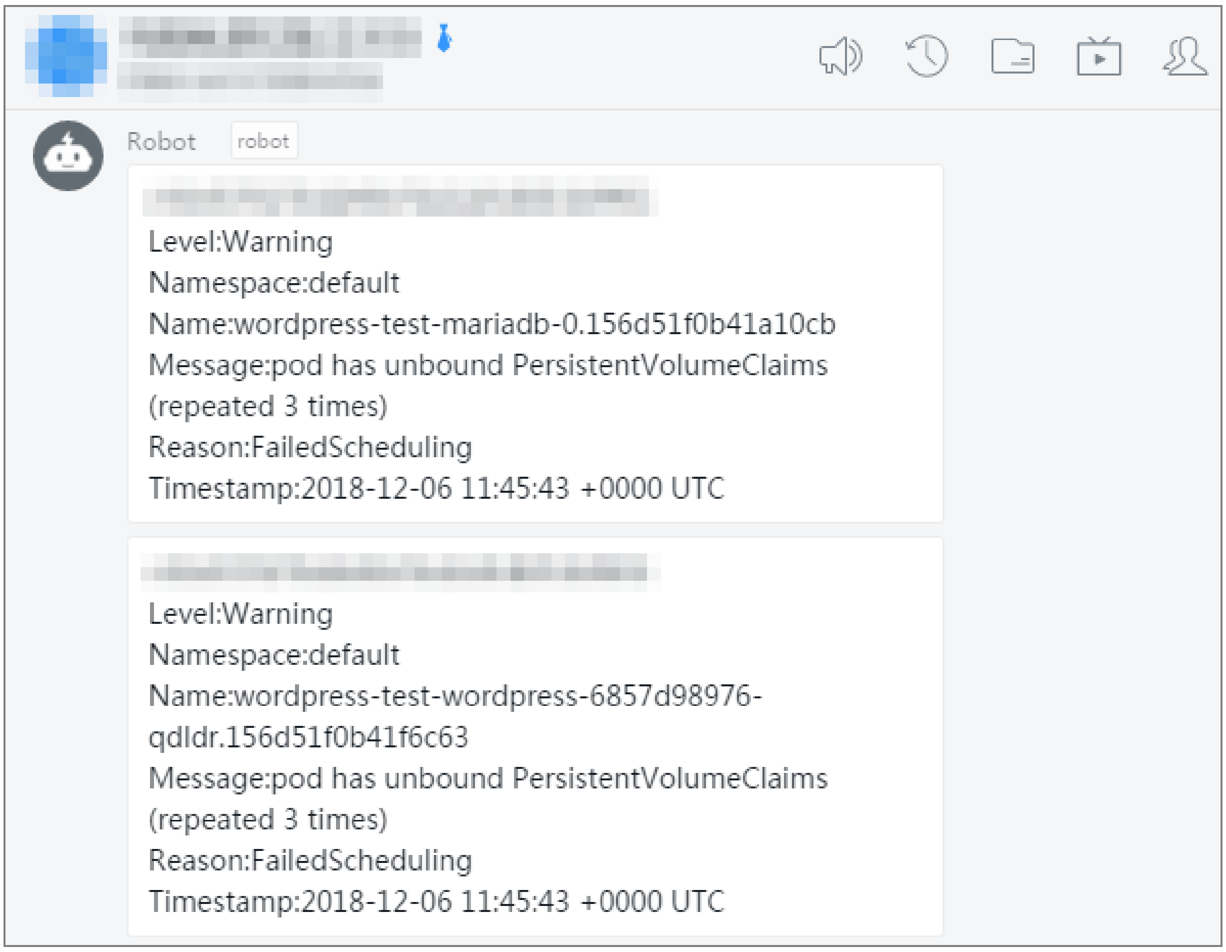
On the **Cluster List** page, click **Dashboard** at the right of the cluster. On the **Dashboard**, select **kube-system** from the drop-down list of **Namespace**, and click **Deployments** in the left-side navigation pane. The deployed eventer is displayed.



Name	Labels	Pods	Age	Images
eventer	k8s-app: eventer task: monitoring	1 / 1	12/06/2018, 19:23:25	registry.cn-hangzhou.aliyuncs.com/acs/eve

Result

The eventer takes effect 30 seconds after you complete the deployment. When an event exceeds the threshold level, you receive the following alarm notifications in the DingTalk group.



Level:Warning
Namespace:default
Name:wordpress-test-mariadb-0.156d51f0b41a10cb
Message:pod has unbound PersistentVolumeClaims (repeated 3 times)
Reason:FailedScheduling
Timestamp:2018-12-06 11:45:43 +0000 UTC

Level:Warning
Namespace:default
Name:wordpress-test-wordpress-6857d98976-qdldr.156d51f0b41f6c63
Message:pod has unbound PersistentVolumeClaims (repeated 3 times)
Reason:FailedScheduling
Timestamp:2018-12-06 11:45:43 +0000 UTC

1.12 Security

1.12.1 Security

Authorization

Kubernetes clusters support authorizing RAM users to perform operations on clusters.

For more information, see *Use sub-accounts*.

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.

1.12.2 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 information that will be
incorporated
...
-----
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:zhejiang
Locality Name (eg, city) [Default City]:hangzhou
Organization Name (eg, company) [Default Company Ltd]:alibaba
Organizational Unit Name (eg, section) []:test
Common Name (eg, your name or your server's hostname) []:foo.bar.com
           #you must 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:

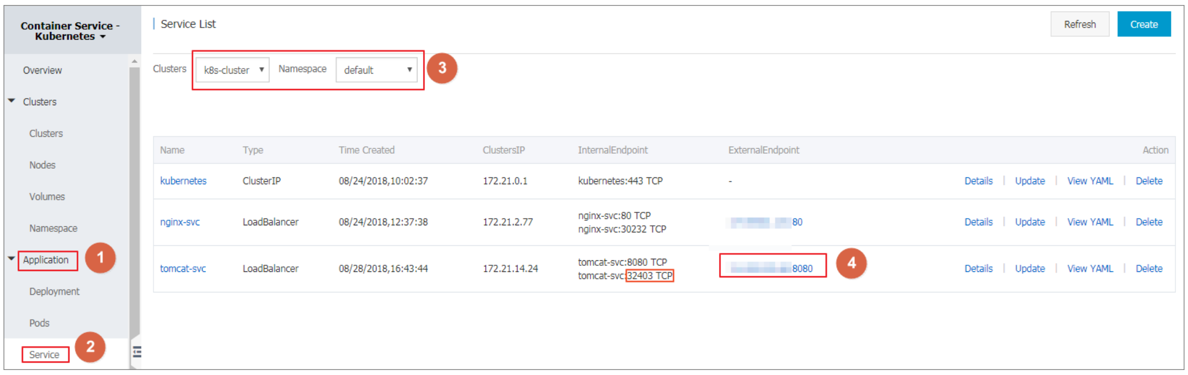
- **Advantages**: The certificate is configured on SLB and it is the external access portal of applications. The access to applications in the cluster still uses the HTTP access method.
- **Disadvantages**: You need to maintain many associations between domain names and their corresponding IP addresses.
- **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*.

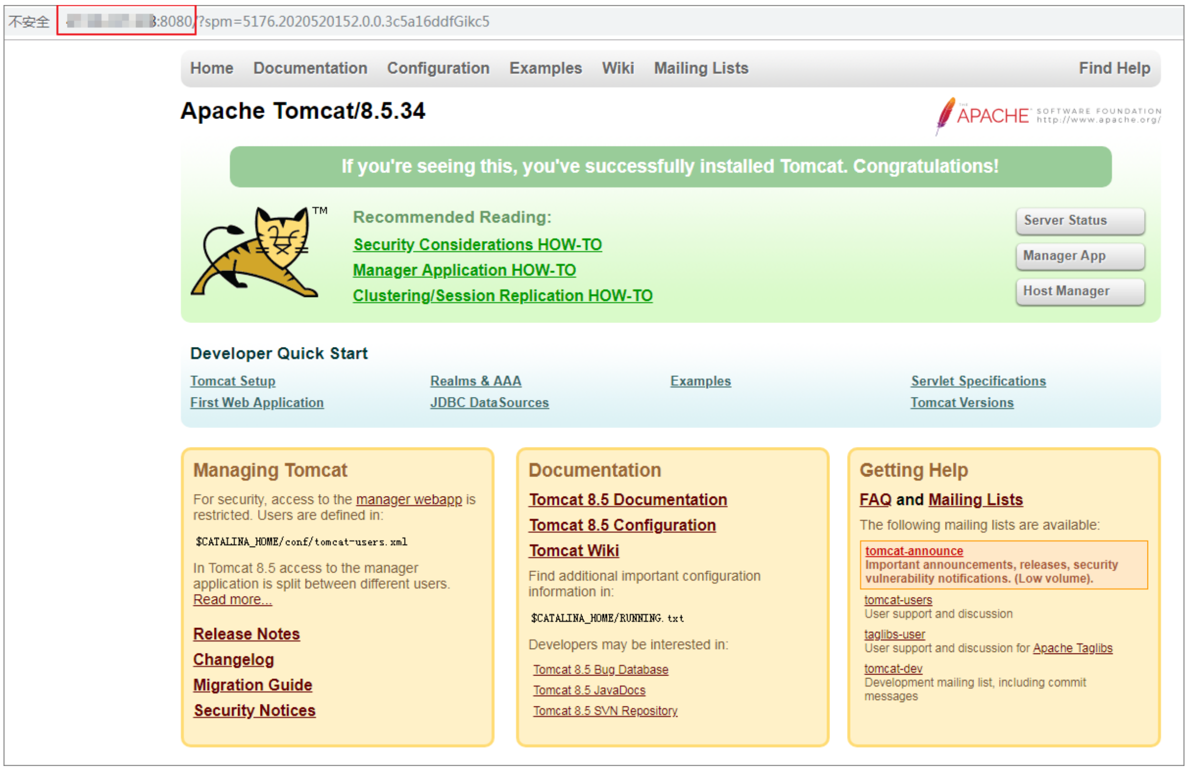
Example

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click **Application** > **Service**, and 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.



Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
kubernetes	ClusterIP	08/24/2018,10:02:37	172.21.0.1	kubernetes:443 TCP	-	Details | Update | View YAML | Delete
nginx-svc	LoadBalancer	08/24/2018,12:37:38	172.21.2.77	nginx-svc:80 TCP nginx-svc:30232 TCP	[redacted]:80	Details | Update | View YAML | Delete
tomcat-svc	LoadBalancer	08/28/2018,16:43:44	172.21.14.24	tomcat-svc:8080 TCP tomcat-svc:32403 TCP	[redacted]:8080	Details | Update | View YAML | Delete

3. By clicking the external endpoint, you can access the Tomcat application through `IP:Port`.


不安全 [redacted]:8080/?spm=5176.2020520152.0.0.3c5a16ddfGikc5
Home
Documentation
Configuration
Examples
Wiki
Mailing Lists
Find Help
APACHE SOFTWARE FOUNDATION http://www.apache.org/
TM
Server Status
Manager App
Host Manager

Apache Tomcat/8.5.34

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

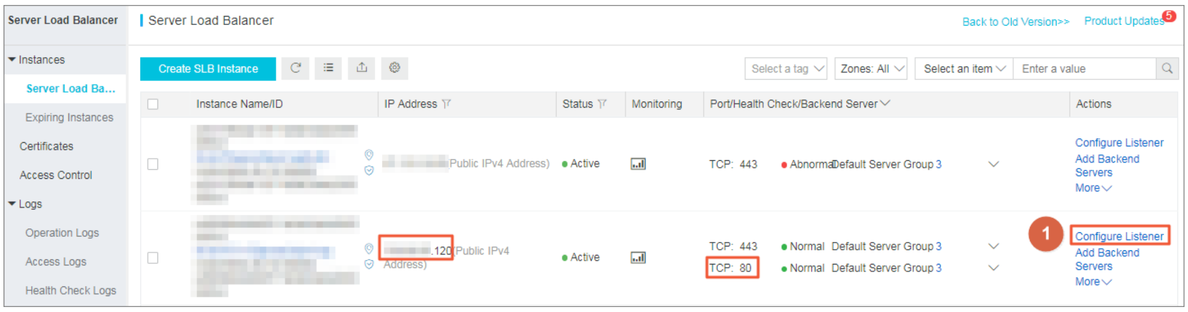
Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 8.5 access to the manager application is split between different users. Read more...
Release Notes
Changelog
Migration Guide
Security Notices

Documentation
Tomcat 8.5 Documentation
Tomcat 8.5 Configuration
Tomcat Wiki
Find additional important configuration information in:
$CATALINA_HOME/RUNNING.txt
Developers may be interested in:
Tomcat 8.5 Bug Database
Tomcat 8.5 JavaDocs
Tomcat 8.5 SVN Repository

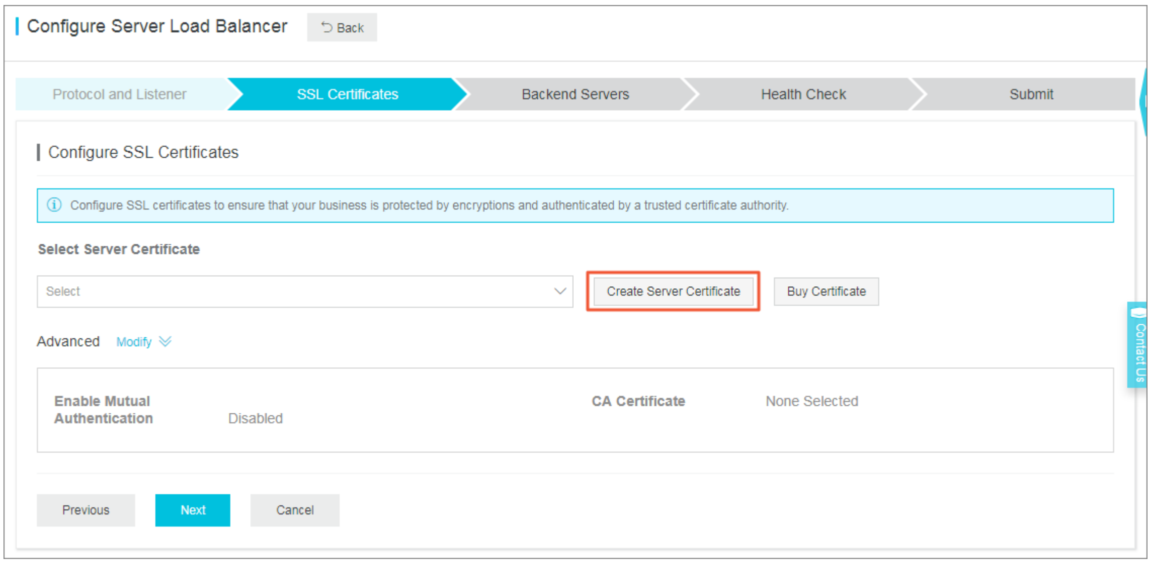
Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).
tomcat-users
User support and discussion
taglibs-user
User support and discussion for Apache Taglibs
tomcat-dev
Development mailing list, including commit messages

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.



	Instance Name/ID	IP Address	Status	Monitoring	Port/Health Check/Backend Server	Actions
	[redacted]	[redacted] Public IPv4 Address)	● Active		TCP: 443 ● Abnormal Default Server Group 3	Configure Listener Add Backend Servers More
	[redacted]	[redacted]120 Public IPv4 Address)	● Active		TCP: 443 ● Normal Default Server Group 3 TCP: 80 ● Normal Default Server Group 3	Configure Listener Add Backend Servers More

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**.

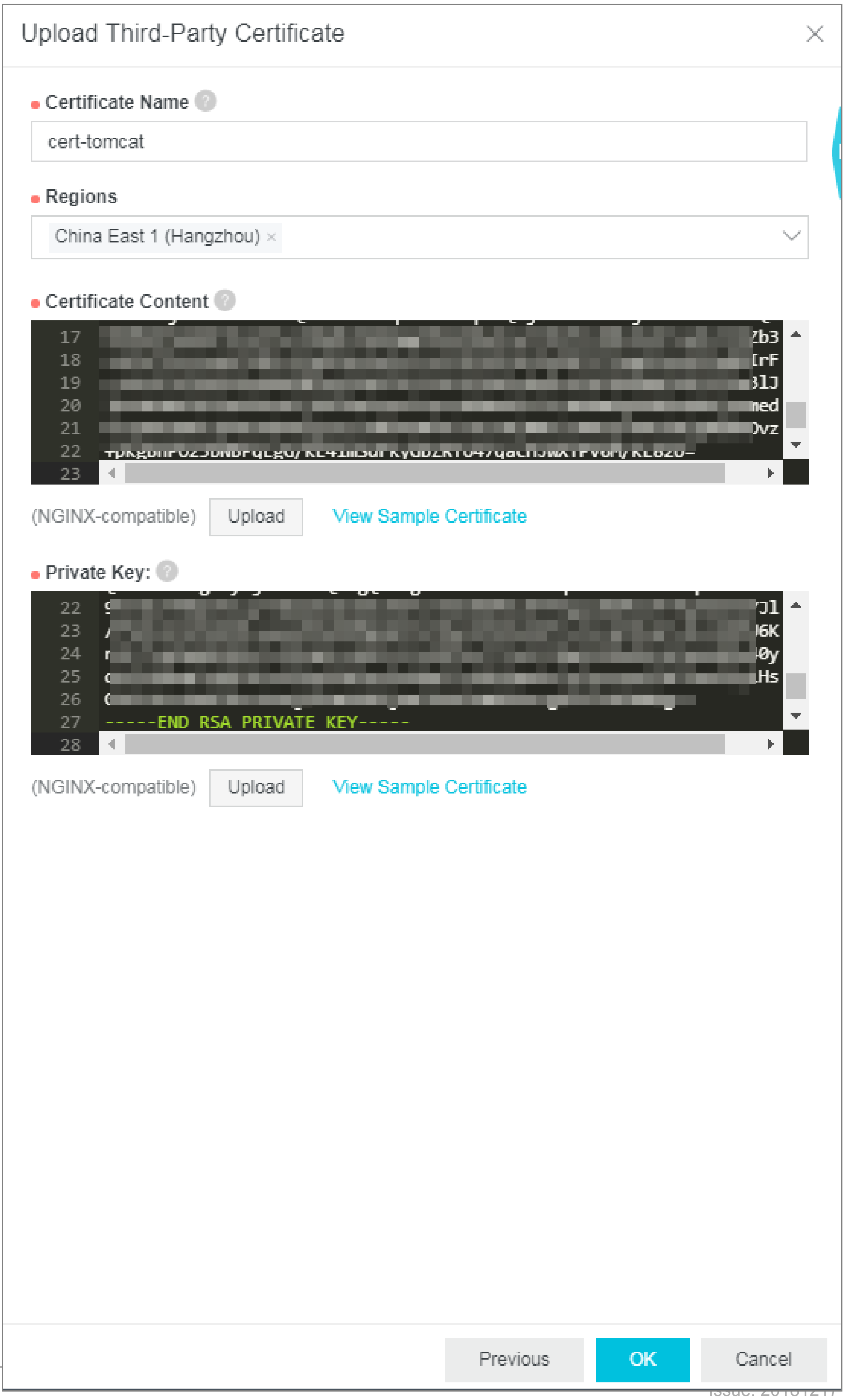


Configure SSL certificates to ensure that your business is protected by encryptions and authenticated by a trusted certificate authority.

Select Server Certificate: Select

Enable Mutual Authentication	Disabled	CA Certificate	None Selected

   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

```
[redacted]
```

(NGINX-compatible) Upload View Sample Certificate

Private Key:

```
[redacted]
-----END RSA PRIVATE KEY-----
```

(NGINX-compatible) Upload View Sample Certificate

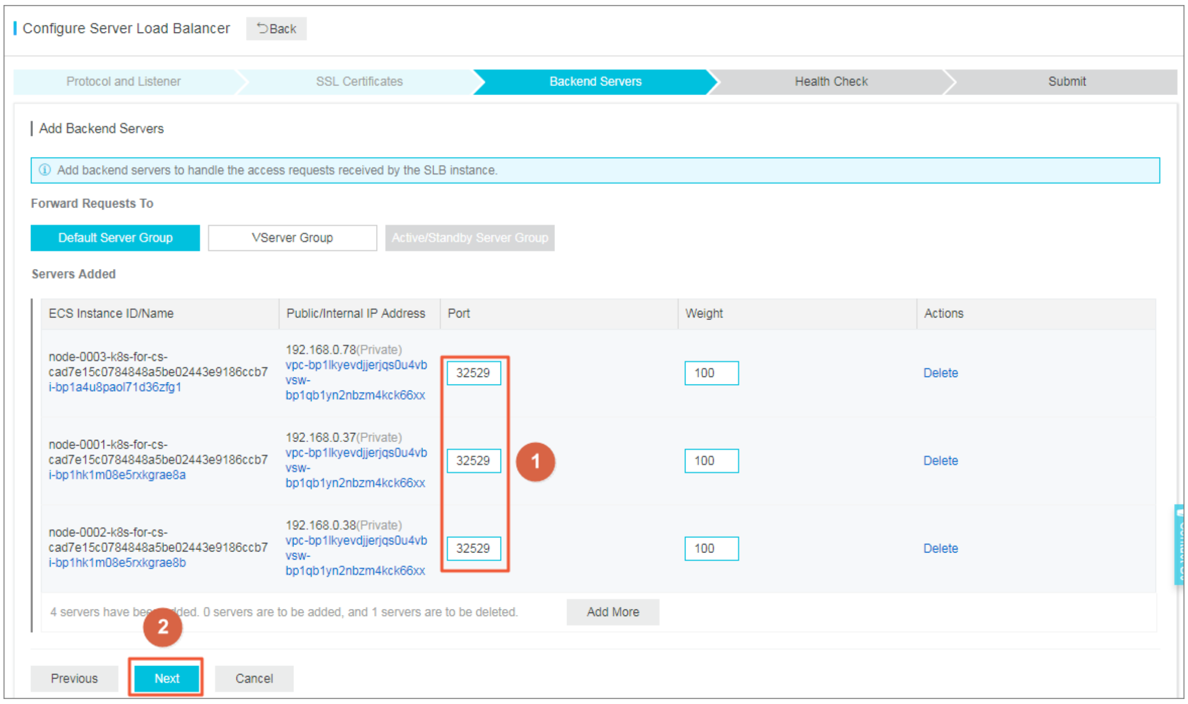
d. From the **Select Server Certificate** drop-down list, select the created server certificate.

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.


Configure Server Load Balancer Back

Add Backend Servers

Add backend servers to handle the access requests received by the SLB instance.

Forward Requests To

Servers Added

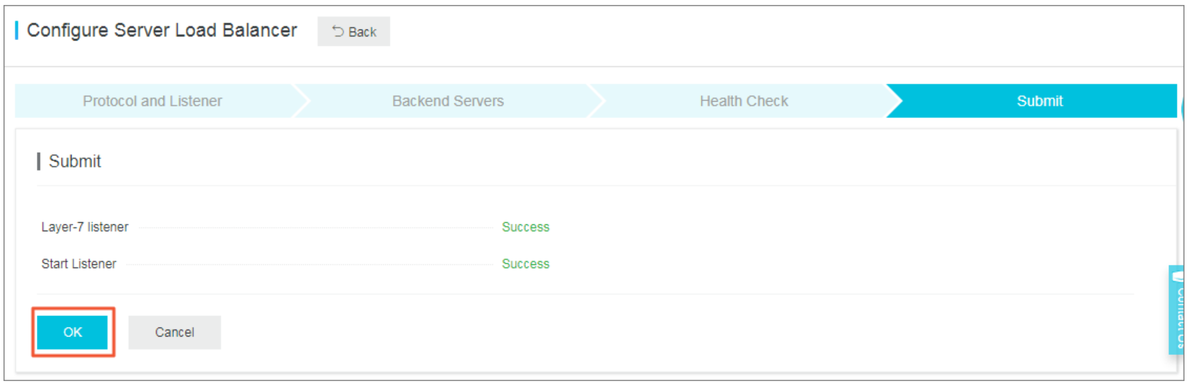
ECS Instance ID/Name	Public/Internal IP Address	Port	Weight	Actions
node-0003-k8s-for-cs-cad7e15c0784848a5be02443e9186ccb7 i-bp1a4u8paol71d36zfg1	192.168.0.78(Private) vpc-bp1lkyevdjjerjqs0u4vb vsw-bp1qb1yn2nbzm4kck66xx	32529	100	Delete
node-0001-k8s-for-cs-cad7e15c0784848a5be02443e9186ccb7 i-bp1hk1m08e5rxkgrae8a	192.168.0.37(Private) vpc-bp1lkyevdjjerjqs0u4vb vsw-bp1qb1yn2nbzm4kck66xx	32529	100	Delete
node-0002-k8s-for-cs-cad7e15c0784848a5be02443e9186ccb7 i-bp1hk1m08e5rxkgrae8b	192.168.0.38(Private) vpc-bp1lkyevdjjerjqs0u4vb vsw-bp1qb1yn2nbzm4kck66xx	32529	100	Delete

4 servers have bee[illegible]dded. 0 servers are to be added, and 1 servers are to be deleted.

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**.


Configure Server Load Balancer Back

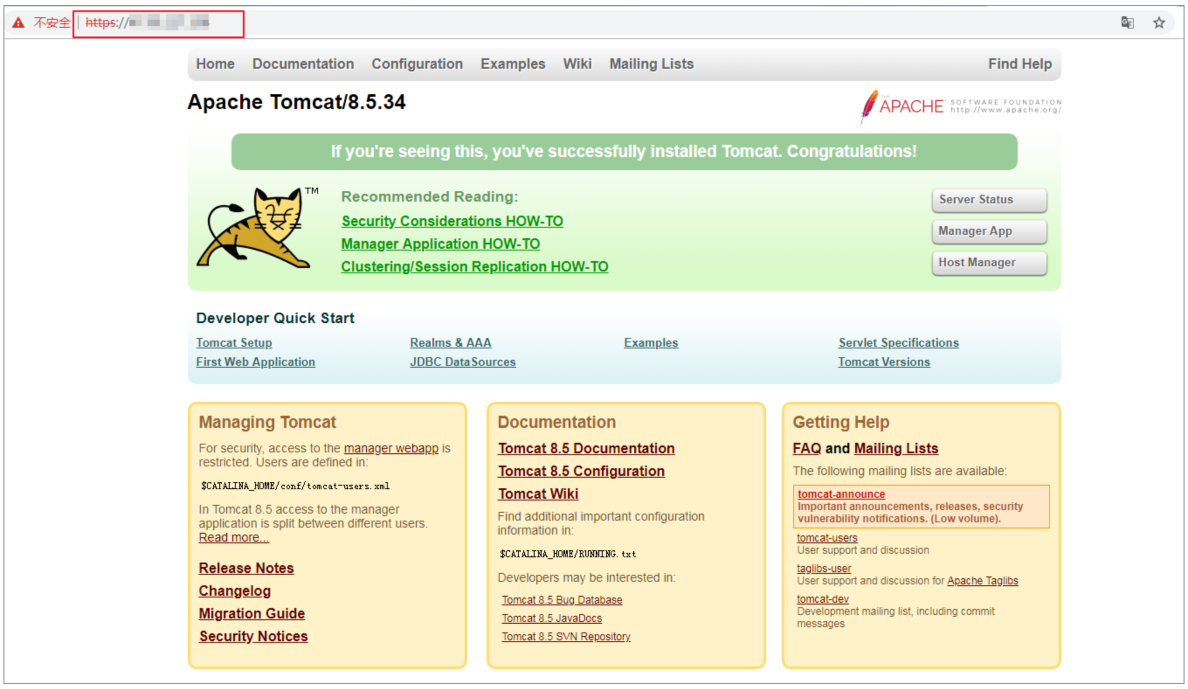
Submit

Layer-7 listener Success

Start Listener Success

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.


不安全 | https://[redacted]

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/8.5.34

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Server Status
Manager App
Host Manager

Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 8.5 access to the manager application is split between different users. Read more...
Release Notes
Changelog
Migration Guide
Security Notices

Documentation
Tomcat 8.5 Documentation
Tomcat 8.5 Configuration
Tomcat Wiki
Find additional important configuration information in:
$CATALINA_HOME/RUNNING.txt
Developers may be interested in:
Tomcat 8.5 Bug Database
Tomcat 8.5 JavaDocs
Tomcat 8.5 SVN Repository

Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).
tomcat-users
User support and discussion
taglibs-user
User support and discussion for Apache Taglibs
tomcat-dev
Development mailing list, including commit messages

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.

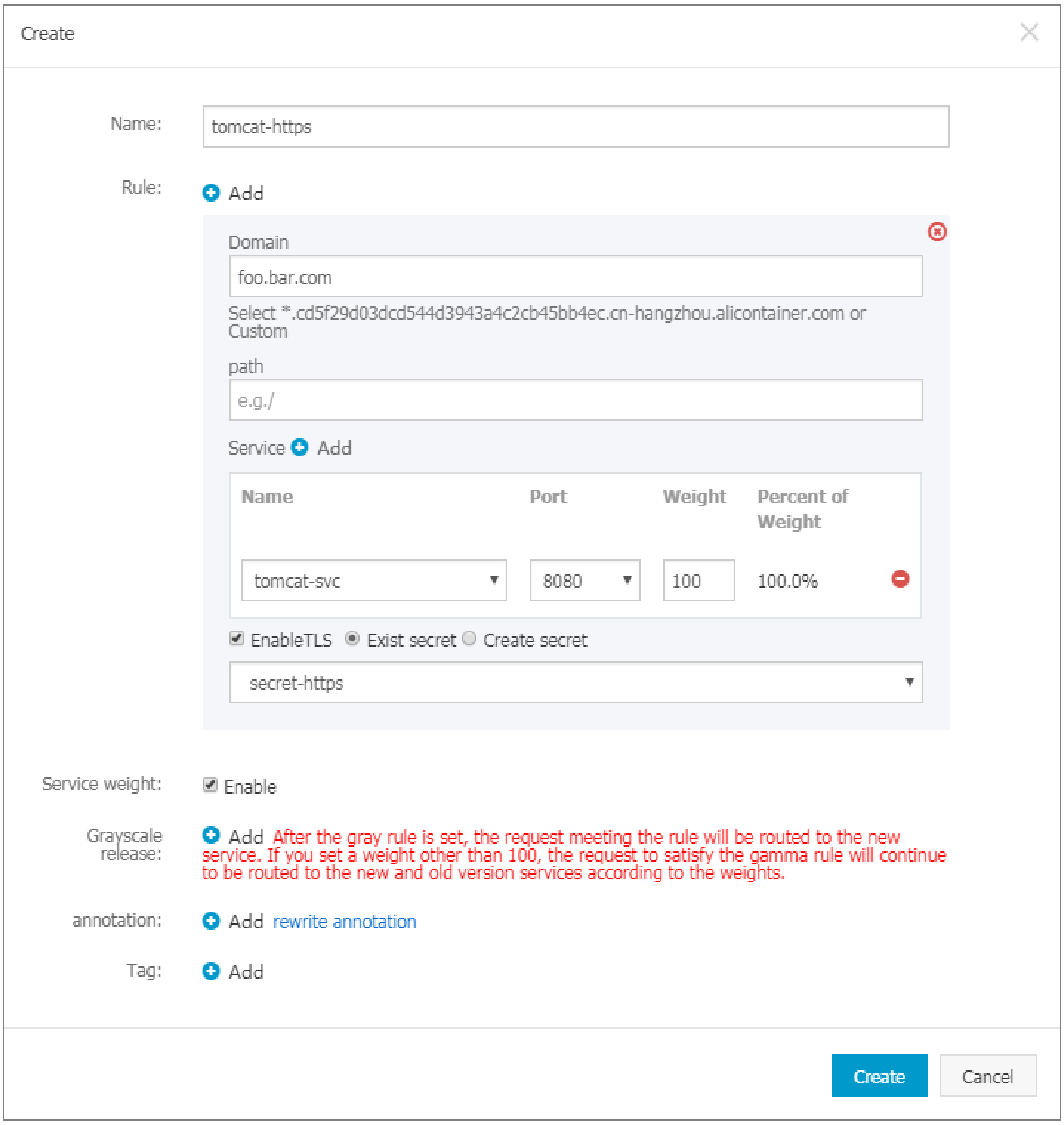
   Note:
   You must set the domain name properly. Otherwise, you will encounter exceptions when accessing the application through HTTPS.

   ```
   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, click **Application** > **Ingress**, 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 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.


Create

Name: tomcat-https

Rule: Add

Domain
foo.bar.com
Select *.cd5f29d03dcd544d3943a4c2cb45bb4ec.cn-hangzhou.alicontainer.com or Custom

path
e.g./

Service Add

Name	Port	Weight	Percent of Weight
tomcat-svc	8080	100	100.0%

[✓ EnableTLS] [✓ Exist secret] Create secret
secret-https

Service weight: [✓ Enable]

Grayscale release: Add After the gray rule is set, the request meeting the rule will be routed to the new service. If you set a weight other than 100, the request to satisfy the gamma rule will continue to be routed to the new and old version services according to the weights.

annotation: Add rewrite annotation

Tag: Add

Create Cancel

You can also use a YAML file to create an Ingress. In this example, the YAML sample file is as follows:

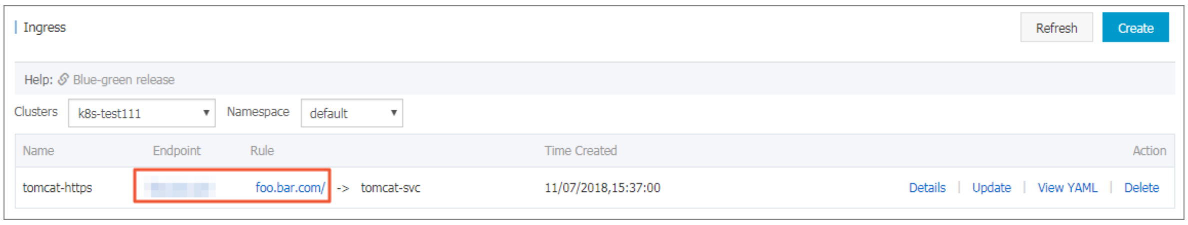
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
   name: tomcat-https
spec:
  tls:
  - hosts:
    - foo.bar.com
    secretName: secret-https
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /
        backend:
          serviceName: tomcat-svc
          servicePort: 8080
```

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.
```



| Ingress

Refresh Create

Help: Blue-green release

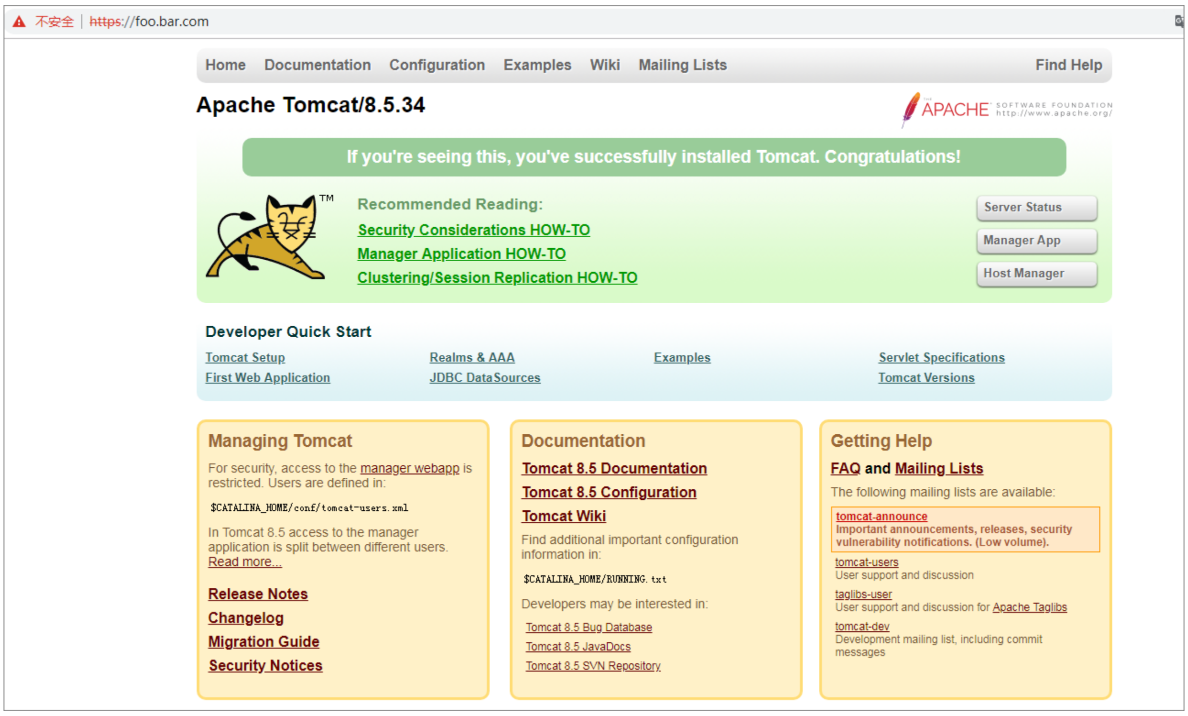
Clusters k8s-test111 Namespace default

Name	Endpoint	Rule	Time Created	Action
tomcat-https	[redacted]	foo.bar.com/ -> tomcat-svc	11/07/2018,15:37:00	Details | Update | View YAML | Delete

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.


不安全 | https://foo.bar.com

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/8.5.34

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Server Status
Manager App
Host Manager

Developer Quick Start

Tomcat Setup	Realms & AAA	Examples	Servlet Specifications
First Web Application	JDBC DataSources		Tomcat Versions

Managing Tomcat

For security, access to the manager webapp is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 8.5 access to the manager application is split between different users. Read more...

Release Notes
Changelog
Migration Guide
Security Notices

Documentation

Tomcat 8.5 Documentation
Tomcat 8.5 Configuration
Tomcat Wiki

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

Tomcat 8.5 Bug Database
Tomcat 8.5 JavaDocs
Tomcat 8.5 SVN Repository

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).

tomcat-users
User support and discussion

taglibs-user
User support and discussion for Apache Taglibs

tomcat-dev
Development mailing list, including commit messages

1.13 Manage a release

1.13.1 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

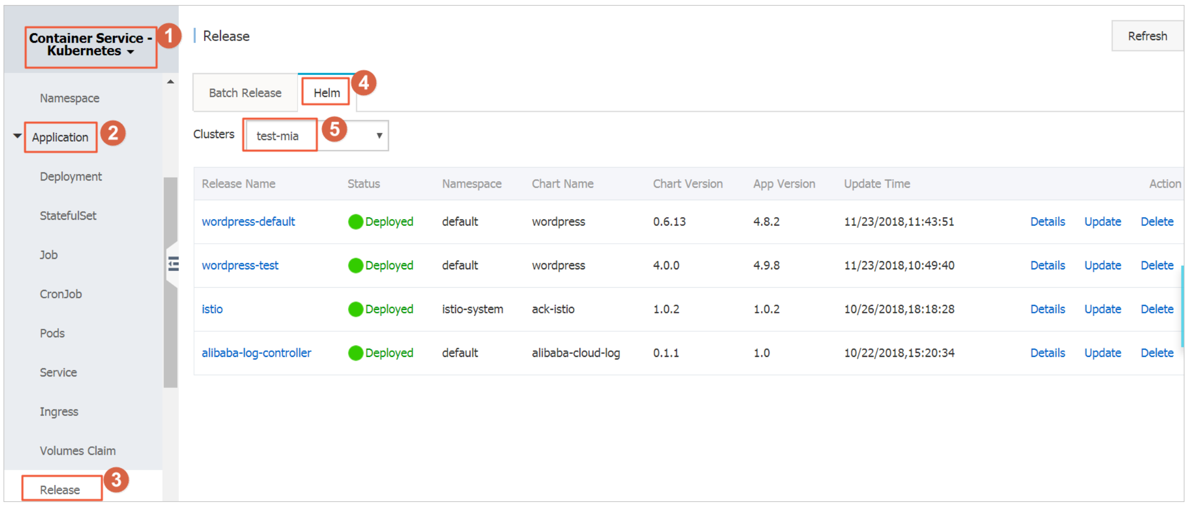
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have installed a Helm application by using the App Catalog function or Service Catalog function. For more information, see *Simplify Kubernetes application deployment by using Helm*. In this topic, 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.

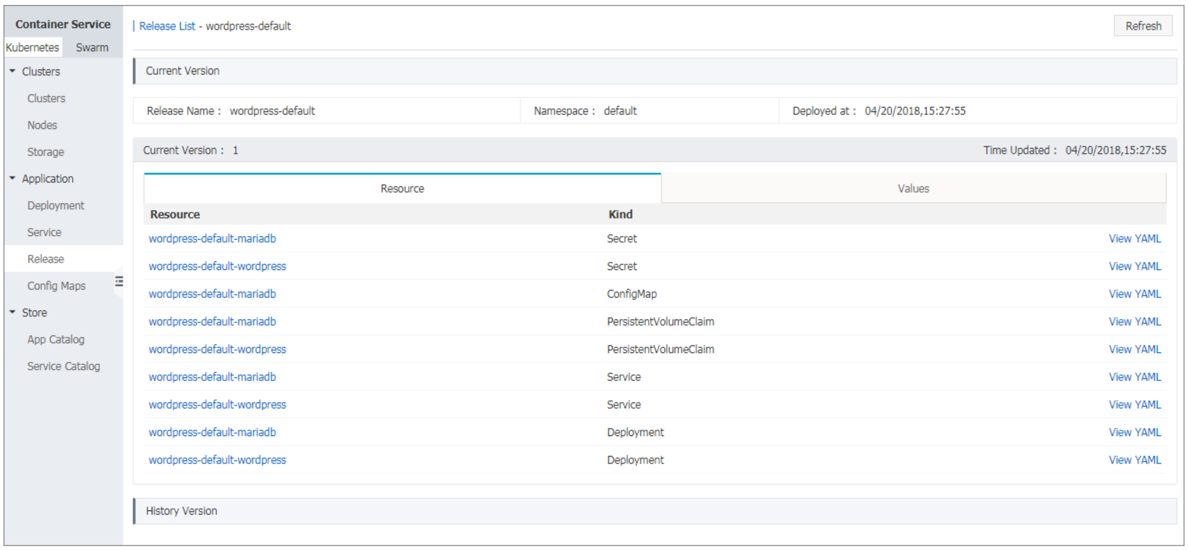


Release Name	Status	Namespace	Chart Name	Chart Version	App Version	Update Time	Action		
wordpress-default	● Deployed	default	wordpress	0.6.13	4.8.2	11/23/2018,11:43:51	Details	Update	Delete
wordpress-test	● Deployed	default	wordpress	4.0.0	4.9.8	11/23/2018,10:49:40	Details	Update	Delete
istio	● Deployed	istio-system	ack-istio	1.0.2	1.0.2	10/26/2018,18:18:28	Details	Update	Delete
alibaba-log-controller	● Deployed	default	alibaba-cloud-log	0.1.1	1.0	10/22/2018,15:20:34	Details	Update	Delete

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.

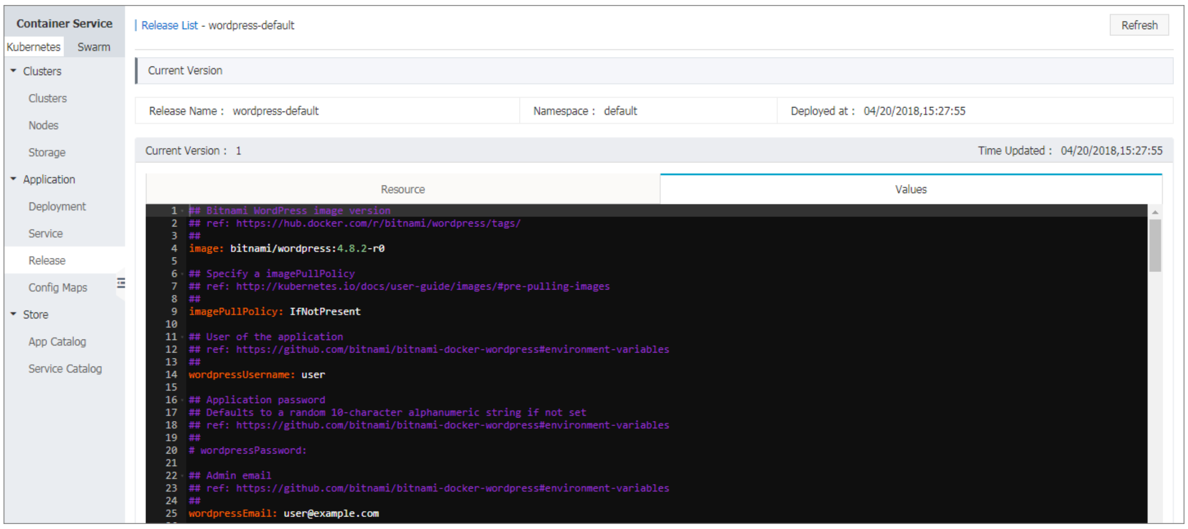


Release Name : wordpress-default | Namespace : default | Deployed at : 04/20/2018,15:27:55

Current Version : 1 | Time Updated : 04/20/2018,15:27:55

Resource	Kind	
wordpress-default-mariadb	Secret	View YAML
wordpress-default-wordpress	Secret	View YAML
wordpress-default-mariadb	ConfigMap	View YAML
wordpress-default-mariadb	PersistentVolumeClaim	View YAML
wordpress-default-wordpress	PersistentVolumeClaim	View YAML
wordpress-default-mariadb	Service	View YAML
wordpress-default-wordpress	Service	View YAML
wordpress-default-mariadb	Deployment	View YAML
wordpress-default-wordpress	Deployment	View YAML

4. Click the **Values** tab to view the release parameters.



Release Name : wordpress-default | Namespace : default | Deployed at : 04/20/2018,15:27:55

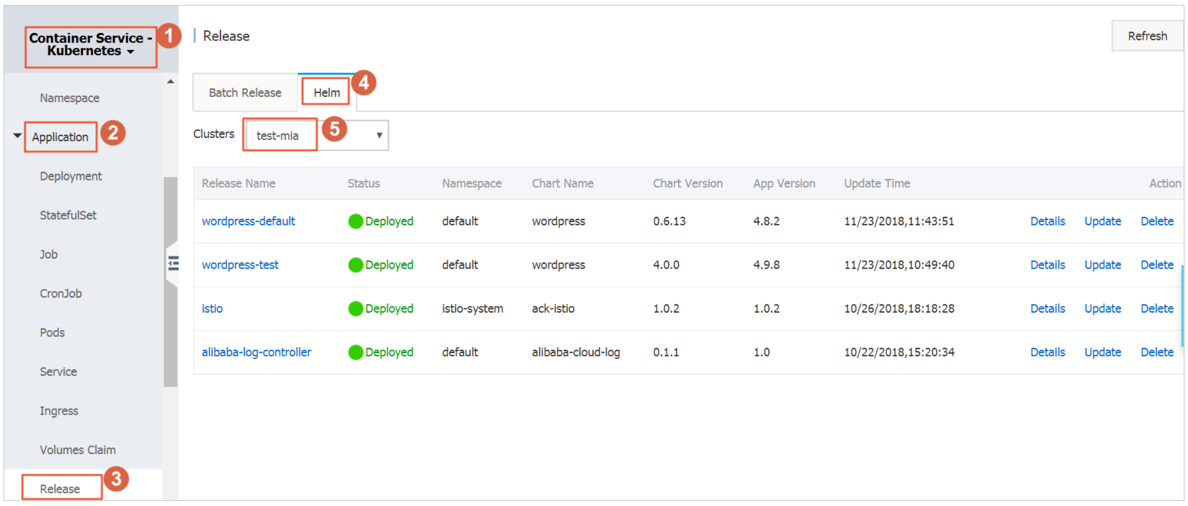
Current Version : 1 | Time Updated : 04/20/2018,15:27:55

```
## Bitnami WordPress image version
## ref: https://hub.docker.com/r/bitnami/wordpress/tags/
##
image: bitnami/wordpress:4.8.2-r0

## Specify a imagePullPolicy
## ref: http://kubernetes.io/docs/user-guide/images/#pre-pulling-images
##
imagePullPolicy: IfNotPresent

## User of the application
## ref: https://github.com/bitnami/bitnami-docker-wordpress#environment-variables
##
wordpressUsername: user

## Application password
## Defaults to a random 10-character alphanumeric string if not set
## ref: https://github.com/bitnami/bitnami-docker-wordpress#environment-variables
##
# wordpressPassword:

## Admin email
## ref: https://github.com/bitnami/bitnami-docker-wordpress#environment-variables
##
wordpressEmail: user@example.com
```

Update a release version

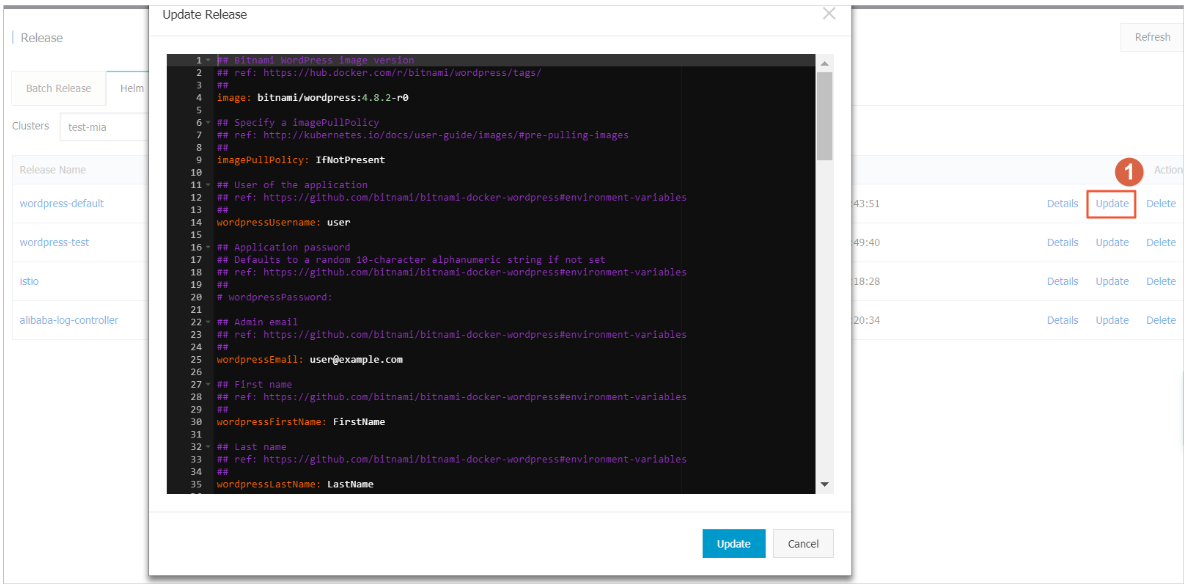
1. Log on to the *Container Service console*.
2. In the left-side navigation pane, select Container Service - Kubernetes. Then, select **Application** > **Release** and click the Helm tab. Select the target cluster from the Clusters drop-down list.

   In the displayed release list, you can view the applications and services released through Helm in the selected cluster.



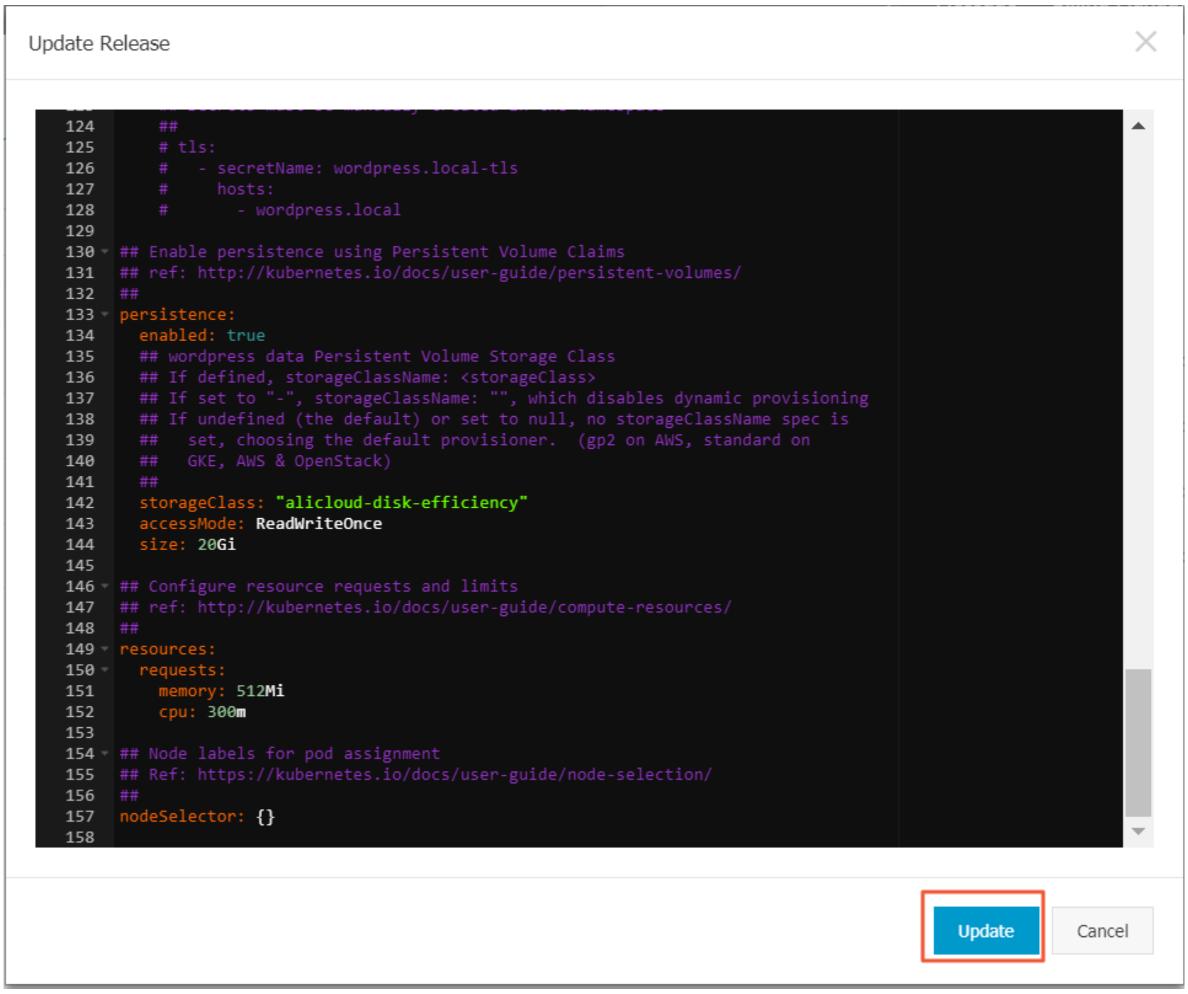
Release Name	Status	Namespace	Chart Name	Chart Version	App Version	Update Time	Action
wordpress-default	● Deployed	default	wordpress	0.6.13	4.8.2	11/23/2018,11:43:51	Details Update Delete
wordpress-test	● Deployed	default	wordpress	4.0.0	4.9.8	11/23/2018,10:49:40	Details Update Delete
istio	● Deployed	istio-system	ack-istio	1.0.2	1.0.2	10/26/2018,18:18:28	Details Update Delete
alibaba-log-controller	● Deployed	default	alibaba-cloud-log	0.1.1	1.0	10/22/2018,15:20:34	Details Update Delete

3. Find your target release (wordpress-default in this example). Click **Update** and the Update Release dialog box appears.



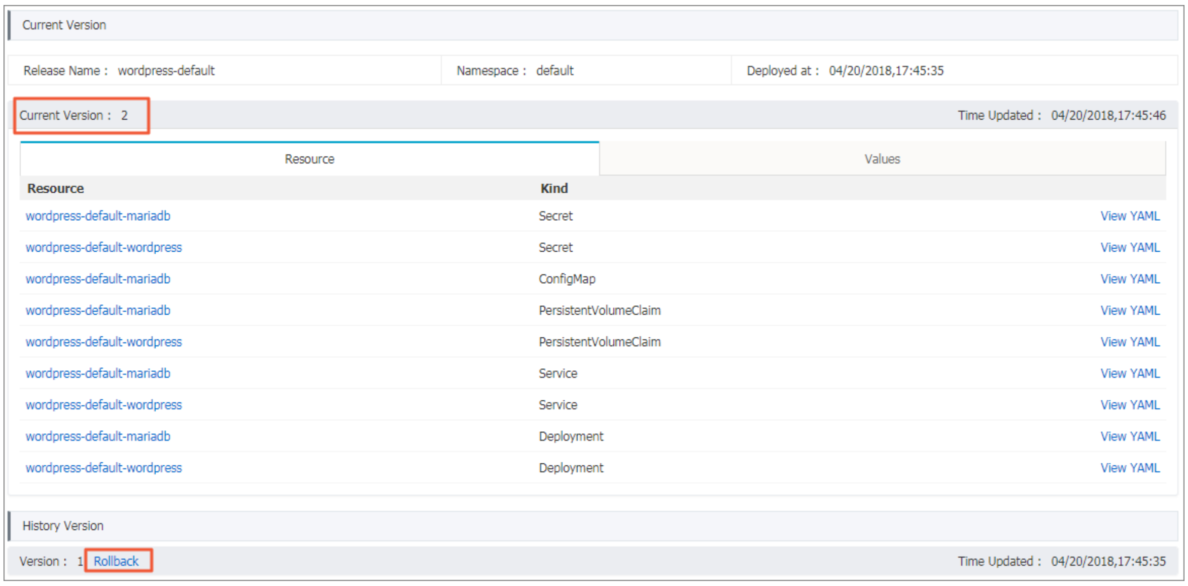
```
## Bitnami WordPress image version
## ref: https://hub.docker.com/r/bitnami/wordpress/tags/
##
image: bitnami/wordpress:4.8.2-r0

## Specify a imagePullPolicy
## ref: http://kubernetes.io/docs/user-guide/images/#pre-pulling-images
##
imagePullPolicy: IfNotPresent

## User of the application
## ref: https://github.com/bitnami/bitnami-docker-wordpress#environment-variables
##
wordpressUsername: user

## Application password
## Defaults to a random 10-character alphanumeric string if not set
## ref: https://github.com/bitnami/bitnami-docker-wordpress#environment-variables
##
# wordpressPassword:

## Admin email
## ref: https://github.com/bitnami/bitnami-docker-wordpress#environment-variables
##
wordpressEmail: user@example.com

## First name
## ref: https://github.com/bitnami/bitnami-docker-wordpress#environment-variables
##
wordpressFirstName: FirstName

## Last name
## ref: https://github.com/bitnami/bitnami-docker-wordpress#environment-variables
##
wordpressLastName: LastName
```

4. Modify the parameters and then click **Update**.



```
    ##
    # tls:
    #   - secretName: wordpress.local-tls
    #     hosts:
    #       - wordpress.local

## Enable persistence using Persistent Volume Claims
## ref: http://kubernetes.io/docs/user-guide/persistent-volumes/
##
persistence:
  enabled: true
  ## wordpress data Persistent Volume Storage Class
  ## If defined, storageClassName: <storageClass>
  ## If set to "-", storageClassName: "", which disables dynamic provisioning
  ## If undefined (the default) or set to null, no storageClassName spec is
  ##   set, choosing the default provisioner.  (gp2 on AWS, standard on
  ##   GKE, AWS & OpenStack)
  ##
  storageClass: "alicloud-disk-efficiency"
  accessMode: ReadWriteOnce
  size: 20Gi

## Configure resource requests and limits
## ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
  requests:
    memory: 512Mi
    cpu: 300m

## Node labels for pod assignment
## Ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

```

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**.



Current Version

Release Name : wordpress-default　Namespace : default　Deployed at : 04/20/2018,17:45:35

Current Version : 2　Time Updated : 04/20/2018,17:45:46

Resource	Kind	
wordpress-default-mariadb	Secret	View YAML
wordpress-default-wordpress	Secret	View YAML
wordpress-default-mariadb	ConfigMap	View YAML
wordpress-default-mariadb	PersistentVolumeClaim	View YAML
wordpress-default-wordpress	PersistentVolumeClaim	View YAML
wordpress-default-mariadb	Service	View YAML
wordpress-default-wordpress	Service	View YAML
wordpress-default-mariadb	Deployment	View YAML
wordpress-default-wordpress	Deployment	View YAML

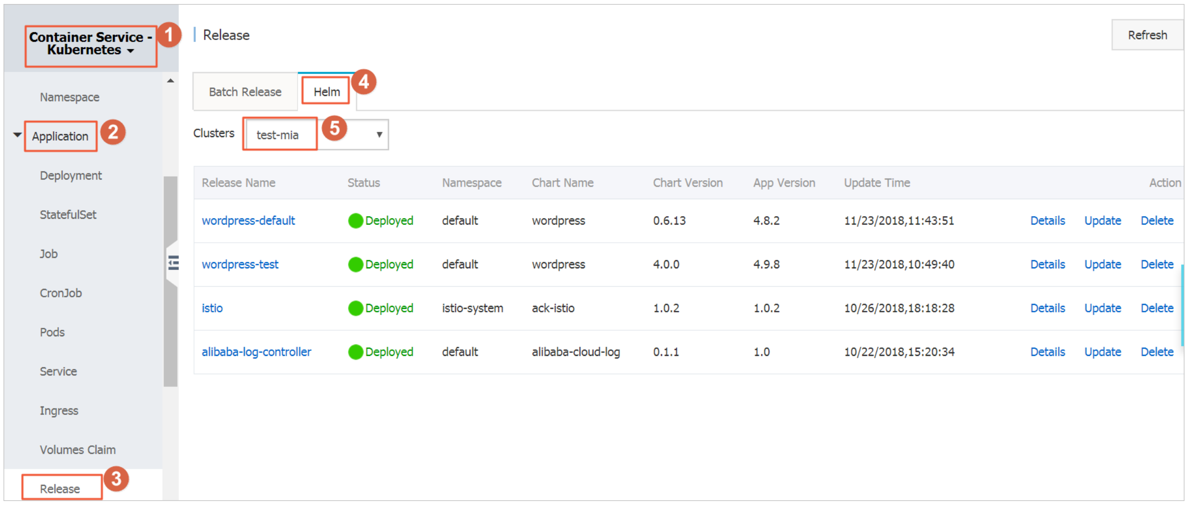
History Version

Version : 1 Rollback　Time Updated : 04/20/2018,17:45:35

Delete a release

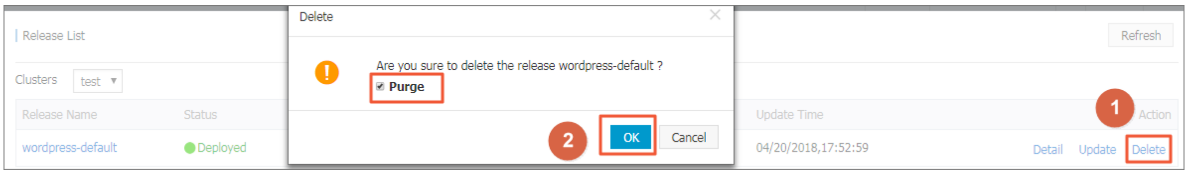
1. Log on to the *Container Service console*.
2. In the left-side navigation pane, select Container Service - Kubernetes. Then, select **Application** > **Release** and click the Helm tab. Select the target cluster from the Clusters drop-down list.

   In the displayed release list, you can view the applications and services released through Helm in the selected cluster.



Release Name	Status	Namespace	Chart Name	Chart Version	App Version	Update Time	Action
wordpress-default	Deployed	default	wordpress	0.6.13	4.8.2	11/23/2018,11:43:51	Details Update Delete
wordpress-test	Deployed	default	wordpress	4.0.0	4.9.8	11/23/2018,10:49:40	Details Update Delete
istio	Deployed	istio-system	ack-istio	1.0.2	1.0.2	10/26/2018,18:18:28	Details Update Delete
alibaba-log-controller	Deployed	default	alibaba-cloud-log	0.1.1	1.0	10/22/2018,15:20:34	Details Update Delete

3. Find your target release (wordpress-default in this example). Click **Delete** and the Delete dialog box appears.



Are you sure to delete the release wordpress-default ?

Release Name	Status	Update Time	Action
wordpress-default	Deployed	04/20/2018,17:52:59	Detail Update Delete

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.

1.13.2 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

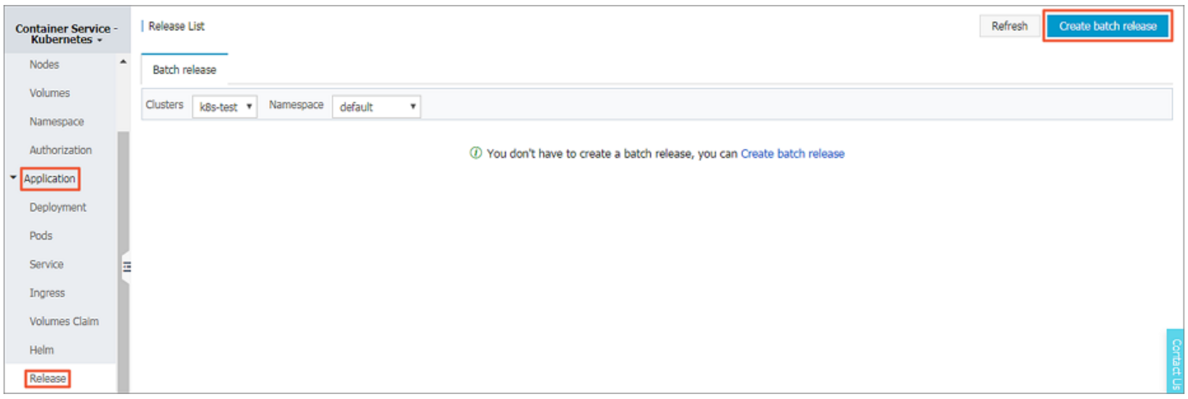
Note:

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

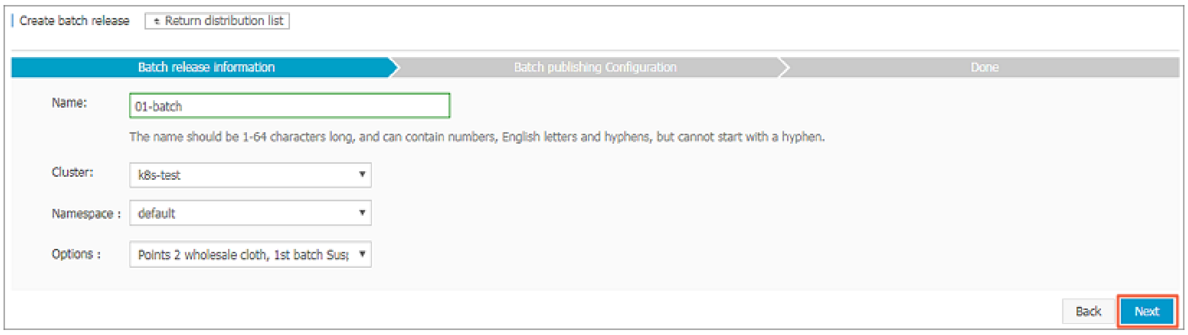
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Release** in the left-side navigation pane. Click **Create batch release** in the upper-right corner.

   Note:
   If the button is gray, you can upgrade the cluster by following the upgrade link.



You don't have to create a batch release, you can Create batch release

3. Configure batch release information, including the application name, cluster, namespace, and options. Click **Next**.



Name: 01-batch

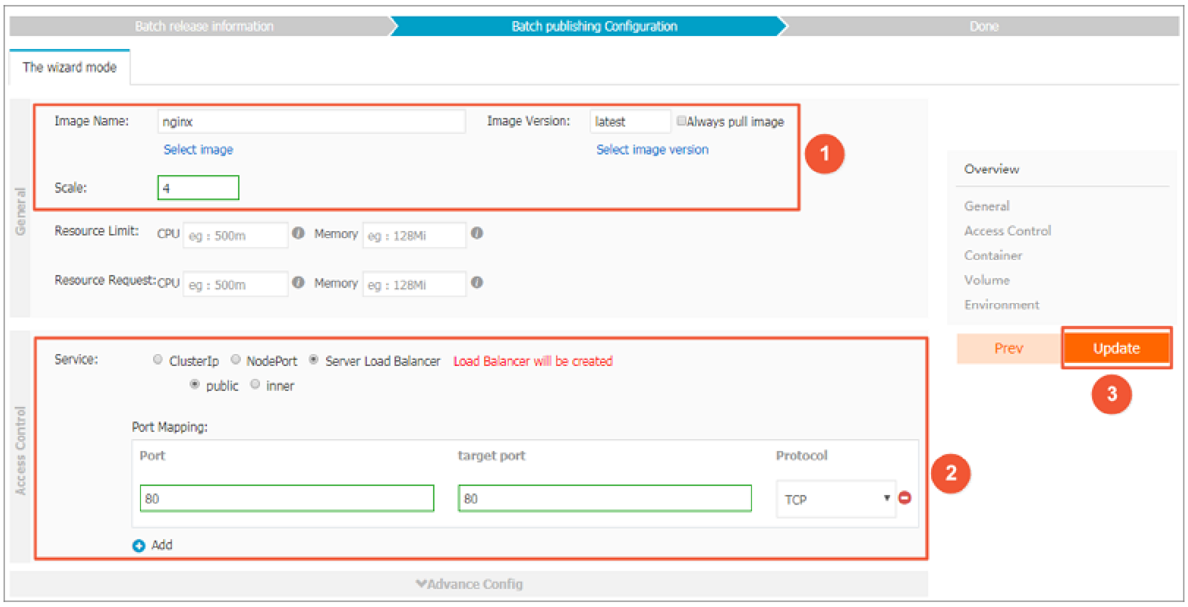
The name should be 1-64 characters long, and can contain numbers, English letters and hyphens, but cannot start with a hyphen.

Cluster: k8s-test

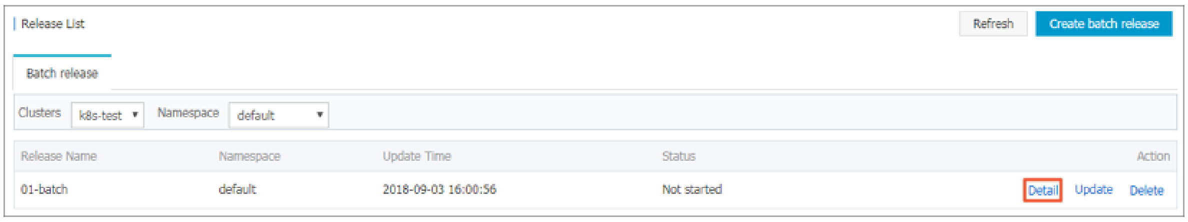
Namespace : default

Options : Points 2 wholesale cloth, 1st batch Sus

4. On the batch publishing configuration page, configure the backend pod and service, and then click **Update** to create an application.

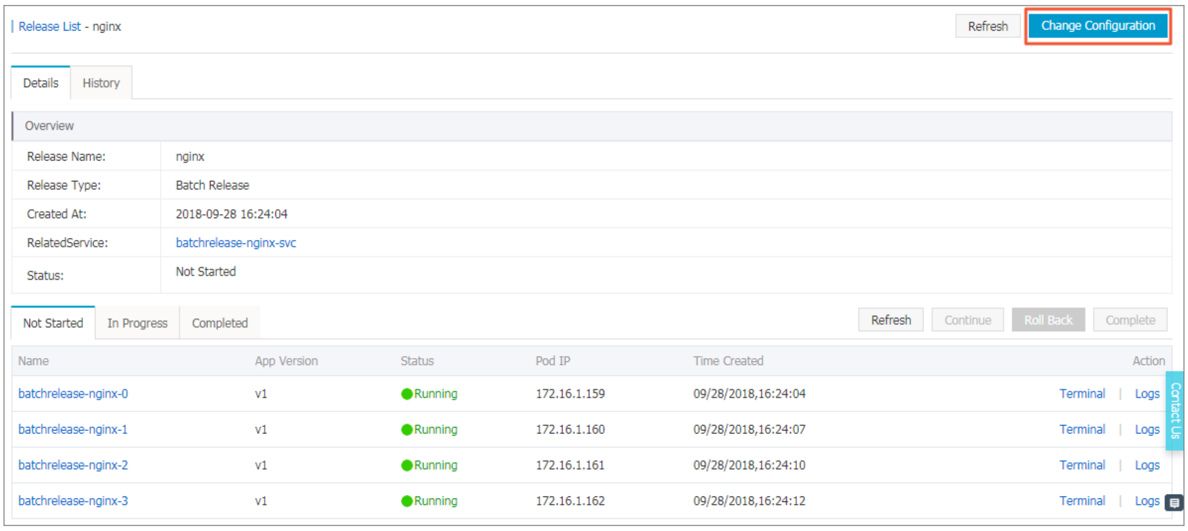


5. Return to the release list, an application is displayed in the **Not started** status. Click **Detail** on the right.



Release Name	Namespace	Update Time	Status	Action
01-batch	default	2018-09-03 16:00:56	Not started	Detail Update Delete

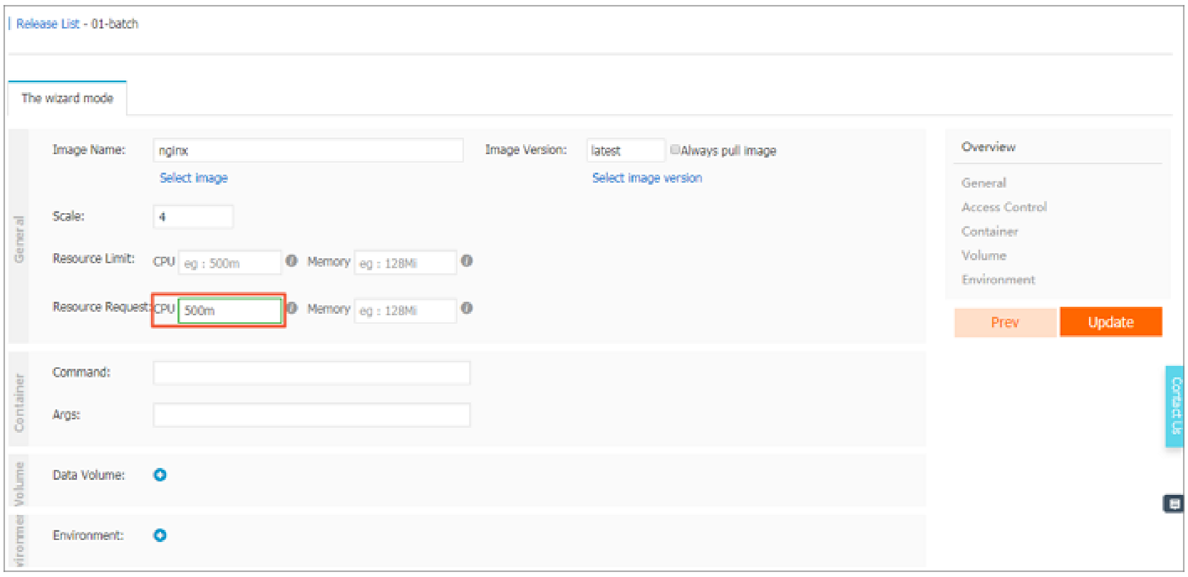
6. On the application detail page, you can view more information. Click **Change Configuration** in the upper-right corner of the page to make a batch release change.



Release Name:	nginx
Release Type:	Batch Release
Created At:	2018-09-28 16:24:04
RelatedService:	batchrelease-nginx-svc
Status:	Not Started

Name	App Version	Status	Pod IP	Time Created	Action
batchrelease-nginx-0	v1	Running	172.16.1.159	09/28/2018,16:24:04	Terminal | Logs
batchrelease-nginx-1	v1	Running	172.16.1.160	09/28/2018,16:24:07	Terminal | Logs
batchrelease-nginx-2	v1	Running	172.16.1.161	09/28/2018,16:24:10	Terminal | Logs
batchrelease-nginx-3	v1	Running	172.16.1.162	09/28/2018,16:24:12	Terminal | Logs

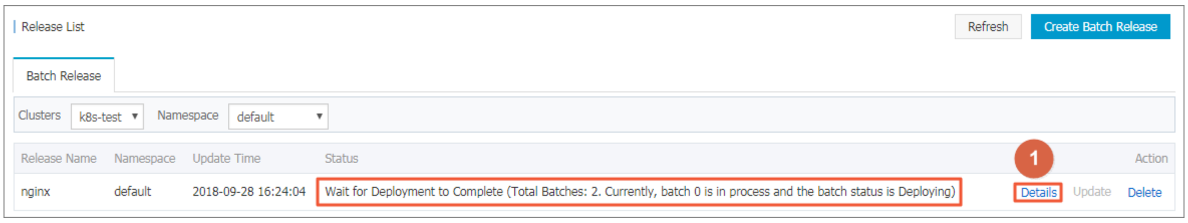
7. Configure changes for the new version of the application, and then click **Update**.


Release List - 01-batch

The wizard mode

Image Name:	nginx
Image Version:	latest ☐ Always pull image
Scale:	4
Resource Limit:	CPU eg : 500m Memory eg : 128Mi
Resource Request:	CPU 500m Memory eg : 128Mi
Command:	
Args:	
Data Volume:	
Environment:	

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**.

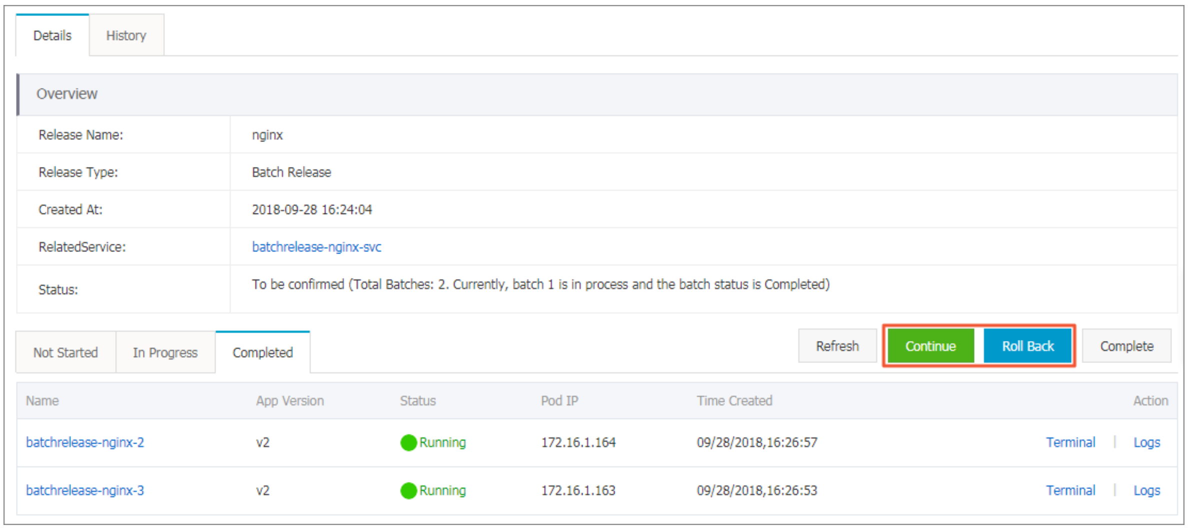

Release List

Batch Release

Clusters k8s-test Namespace default

Release Name	Namespace	Update Time	Status	Action
nginx	default	2018-09-28 16:24:04	Wait for Deployment to Complete (Total Batches: 2. Currently, batch 0 is in process and the batch status is Deploying)	Details Update Delete

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.


Details History

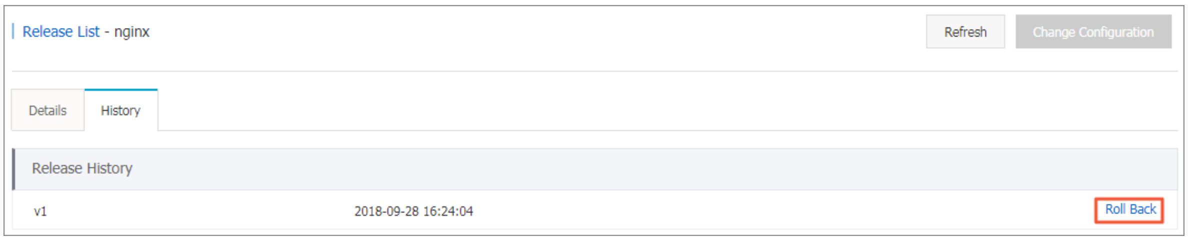
Overview

Release Name:	nginx
Release Type:	Batch Release
Created At:	2018-09-28 16:24:04
RelatedService:	batchrelease-nginx-svc
Status:	To be confirmed (Total Batches: 2. Currently, batch 1 is in process and the batch status is Completed)

Not Started In Progress Completed

Name	App Version	Status	Pod IP	Time Created	Action
batchrelease-nginx-2	v2	● Running	172.16.1.164	09/28/2018,16:26:57	Terminal | Logs
batchrelease-nginx-3	v2	● Running	172.16.1.163	09/28/2018,16:26:53	Terminal | Logs

10. When completing the release, click **History**to roll back to history versions.



Release History

v1 2018-09-28 16:24:04

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.

1.14 Istio

1.14.1 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:

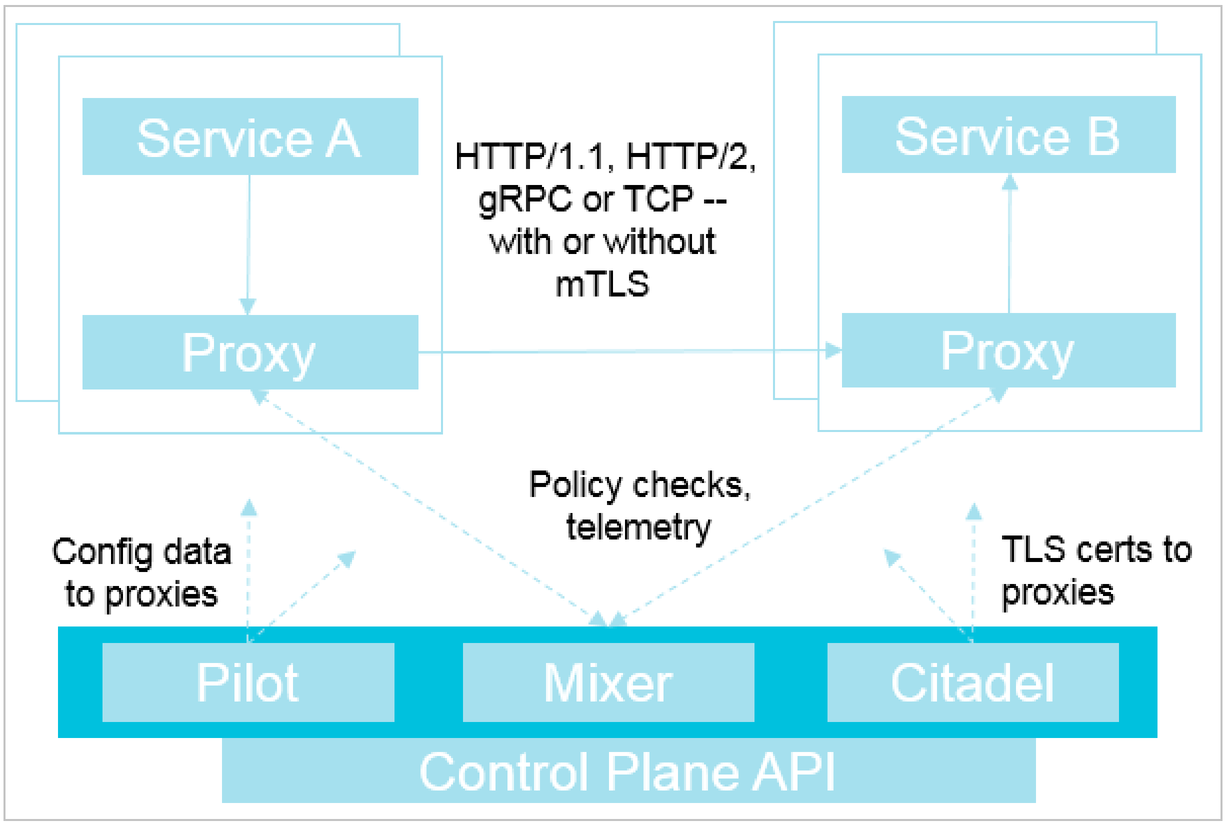
- Traffic management: Controls traffic and API calls between services to enhance the system reliability.
- Authentication and security protection: Provides authentication for services in meshes, and protects the traffic of services to enhance the system security.

- 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 representa tion 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.

1.14.2 Deploy Istio

The distributed application architecture composed of microservices has disadvantages in aspects such as operation and maintenance (O&M), debugging, and security management. To eliminate the disadvantages, you can deploy Istio to create microservice network and to provide load balancing, service-to-service authentication, monitoring, and other functions. Istio provides the functions without requiring any changes to services.

Prerequisites

- — You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have logged on to the Container Service console by using the primary account or by using a sub-account that has been granted sufficient permissions. For example, if the `cluster-admin` permission is granted to a sub-account then Istio can be deployed. Other combinations of permissions are also sufficient. For more information, see *Sub-account Kubernetes permission configuration guide*.

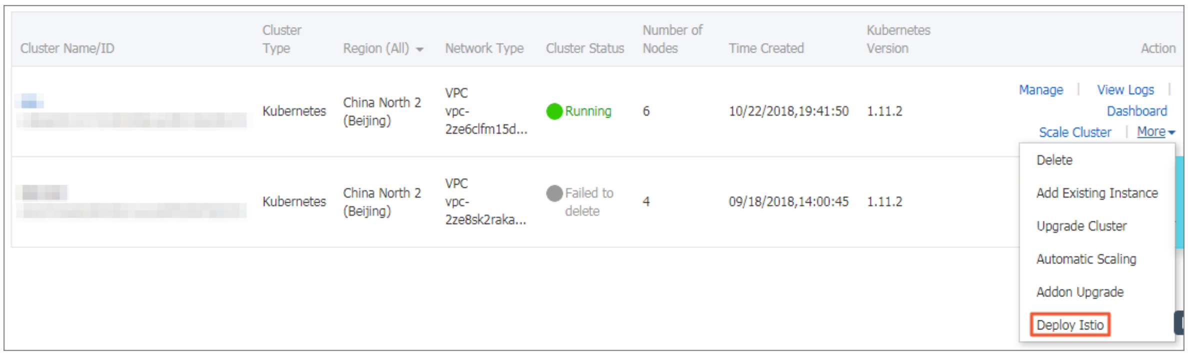
Background information

- Alibaba Cloud Container Service for Kubernetes in versions of 1.10.4 and later support Istio deployment. If your Container Service for Kubernetes is in any version prior to 1.10.4, update the version to 1.10.4 or later.
- To guarantee sufficient resources, the number of Worker nodes in a cluster must be greater than or equal to 3.

Procedure

Deploy Istio through a cluster interface

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click **Clusters**.
3. Select a cluster and then click **More** > **Deploy Istio** in the action column at the right of a cluster.



Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Number of Nodes	Time Created	Kubernetes Version	Action
[redacted]	Kubernetes	China North 2 (Beijing)	VPC vpc-2ze6clfm15d...	● Running	6	10/22/2018,19:41:50	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More
[redacted]	Kubernetes	China North 2 (Beijing)	VPC vpc-2ze8sk2raka...	● Failed to delete	4	09/18/2018,14:00:45	1.11.2	

4. Deploy Istio according to the following information.

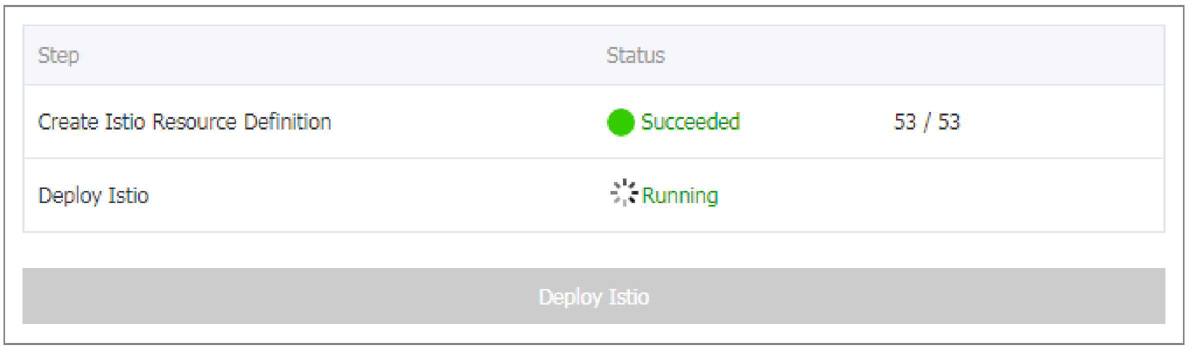
Configuration	Description
Clusters	Target cluster in which Istio is deployed.
Namespace	Namespace in which Istio is deployed.
Release Name	Name of Istio to be released.
Enable Prometheus for metrics/logs collection	Whether to enable Prometheus for metrics/logs collection. Enabled by default.
Enable Grafana for metrics display	Whether to enable Grafana for metrics display. Enabled by default.
Enable automatic Istio Sidecar injection	Whether to enable automatic Istio Sidecar injection . Enabled by default.
Enable the Kiali Visualization Service Mesh	Whether to enable the Kiali Visualization Service Mesh. Disabled by default.

Configuration	Description
	• Username: Set a user name. The default is admin. • Password: Set a password. The default is admin .
Enable Log Service(SLS) and Jaeger	Whether to enable Log Service(SLS) and Jaeger. Disabled by default. Endpoint: Select an address according to the region in which the configured Log Service exists. For more information, see *Service endpoint*. Project: Set the name of the project in which you collect logs. Logstore: Set the name of the Logstore in which you collect logs. AccessKeyID: Set the ID of the AccessKey used to access Log Service. **Note:** Select the AccessKeyID that has permission to access Log Service. AccessKeySecret: Set the AccessKeySecret used to access Log Service.
Pilot Settings	Set the trace sampling percentage in the range of 0 to 100. The default is 1.
Control Egress Traffic	• Permitted Addresses for External Access: range of IP addresses that can be used to directly access services in the Istio service mesh. By default, this field is left blank. Use commas (,) to separate multiple IP address ranges. • Blocked Addresses for External Access: 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. **Note:**

Configuration	Description
	If the settings of these two parameters conflict with each other, the Permitted Addresses for External Access prevails. For example, if an IP address is listed in both IP address ranges that you set for these two parameters, the IP address can be still accessed . That is, the setting of Permitted Addresses for External Access prevails.

5. Click **Deploy Istio** to start deployment.

   At the bottom of the deployment page, you can view the deployment progress and status in real time.

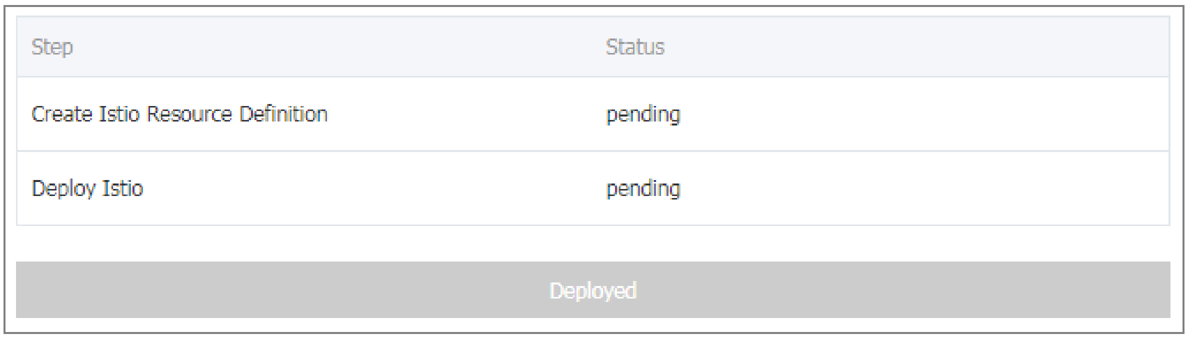


Step	Status	
Create Istio Resource Definition	● Succeeded	53 / 53
Deploy Istio	Running	

Expected results:

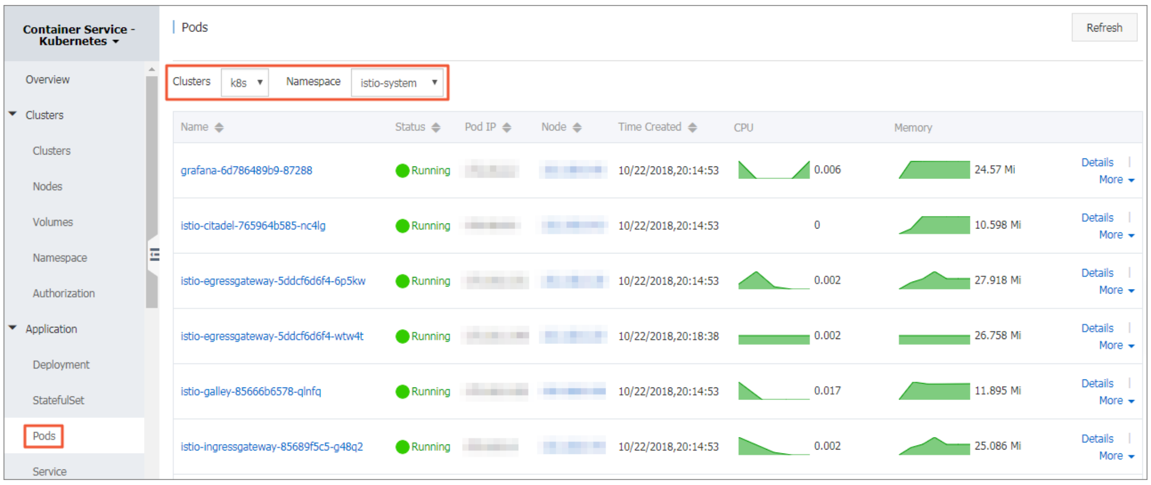
You can view your deployment results in the following ways:

- At the bottom of the **Deploy Istio** page, **Deploy Istio** is changed to **Deployed**.



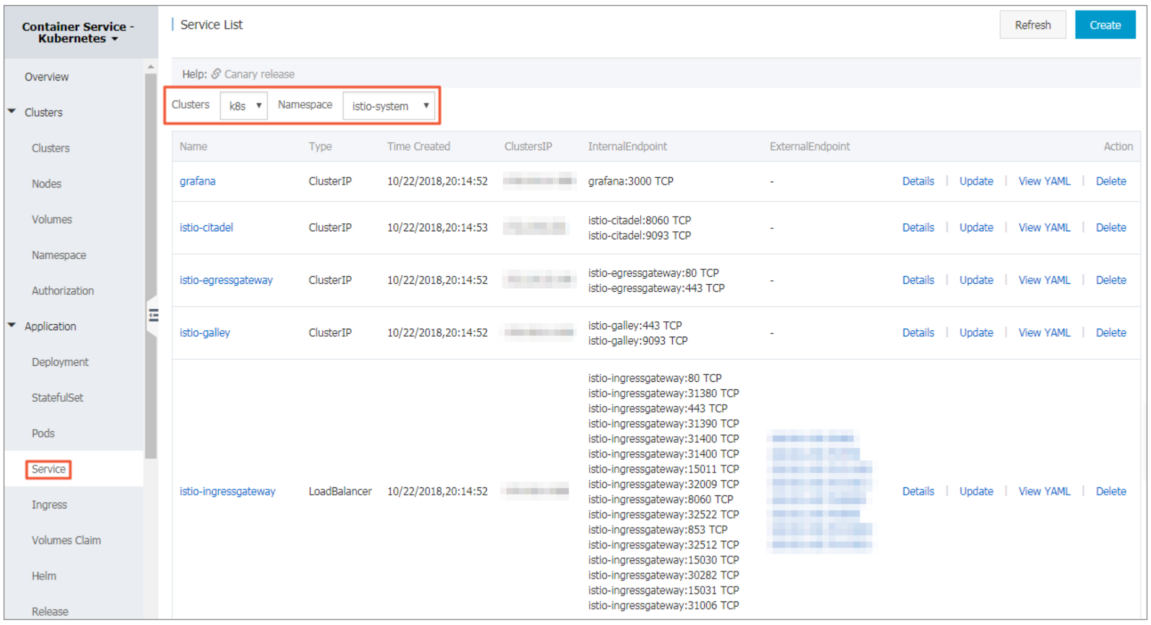
Step	Status
Create Istio Resource Definition	pending
Deploy Istio	pending

- • In the left-side navigation pane, click **Application** > **Pods**.
  - Select the cluster and namespace in which Istio is deployed, and you can see the relevant pods in which Istio is deployed.



Name	Status	Pod IP	Node	Time Created	CPU	Memory	
grafana-6d786489b9-87288	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0.006	24.57 Mi	Details | More
istio-citadel-765964b585-nc4lg	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0	10.598 Mi	Details | More
istio-egressgateway-5ddcf6d6f4-6p5kw	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0.002	27.918 Mi	Details | More
istio-egressgateway-5ddcf6d6f4-wtw4t	Running	[redacted]	[redacted]	10/22/2018,20:18:38	0.002	26.758 Mi	Details | More
istio-galley-85666b6578-qlnfq	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0.017	11.895 Mi	Details | More
istio-ingressgateway-85689f5c5-g48q2	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0.002	25.086 Mi	Details | More

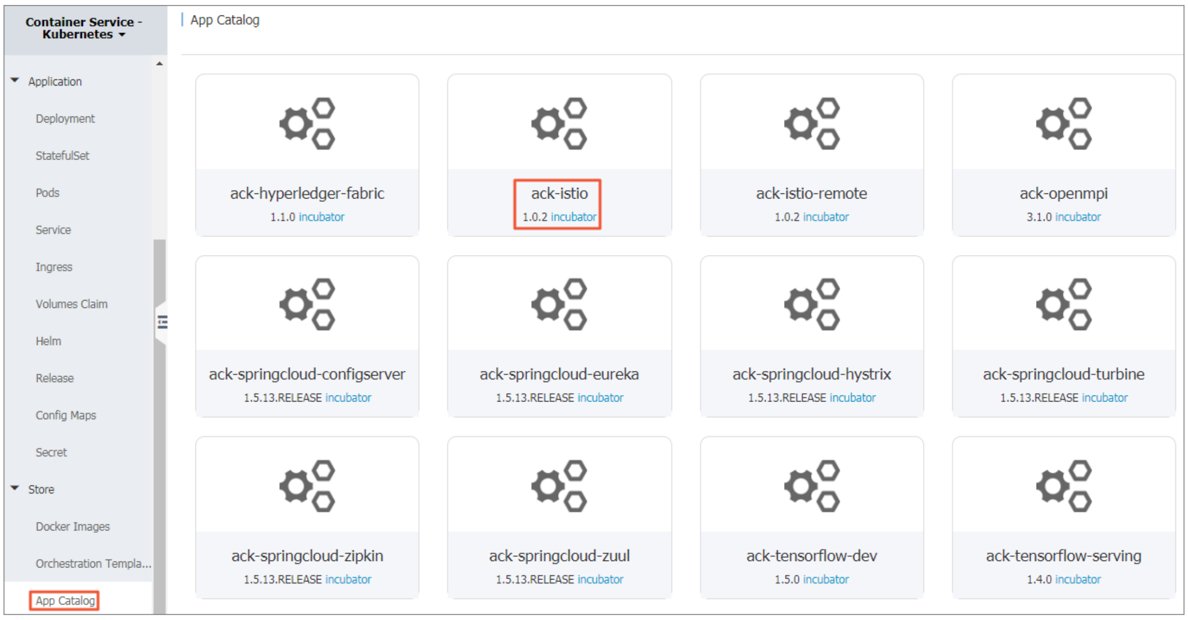
- • In the left-side navigation pane, click **Application** > **Service**.
  - Select the cluster and namespace in which Istio is deployed, and you can see the access addresses provided by the relevant services in which Istio is deployed.



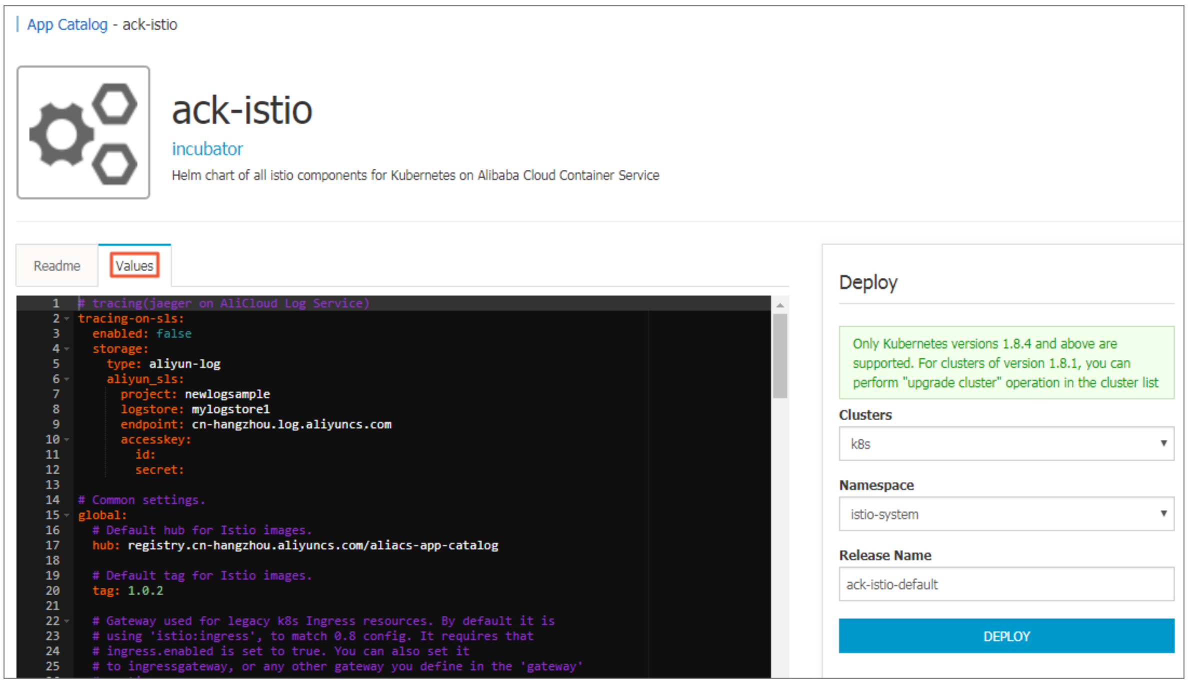
Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
grafana	ClusterIP	10/22/2018,20:14:52	[redacted]	grafana:3000 TCP	-	Details | Update | View YAML | Delete
istio-citadel	ClusterIP	10/22/2018,20:14:53	[redacted]	istio-citadel:8060 TCP istio-citadel:9093 TCP	-	Details | Update | View YAML | Delete
istio-egressgateway	ClusterIP	10/22/2018,20:14:52	[redacted]	istio-egressgateway:80 TCP istio-egressgateway:443 TCP	-	Details | Update | View YAML | Delete
istio-galley	ClusterIP	10/22/2018,20:14:52	[redacted]	istio-galley:443 TCP istio-galley:9093 TCP	-	Details | Update | View YAML | Delete
istio-ingressgateway	LoadBalancer	10/22/2018,20:14:52	[redacted]	istio-ingressgateway:80 TCP istio-ingressgateway:31380 TCP istio-ingressgateway:443 TCP istio-ingressgateway:31390 TCP istio-ingressgateway:31400 TCP istio-ingressgateway:31400 TCP istio-ingressgateway:15011 TCP istio-ingressgateway:32009 TCP istio-ingressgateway:8060 TCP istio-ingressgateway:32522 TCP istio-ingressgateway:853 TCP istio-ingressgateway:32512 TCP istio-ingressgateway:15030 TCP istio-ingressgateway:30282 TCP istio-ingressgateway:15031 TCP istio-ingressgateway:31006 TCP	[redacted]	Details | Update | View YAML | Delete

Use the application catalog to deploy Istio

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click **Store** > **App Catalog**.


Container Service - Kubernetes
App Catalog
Application
Deployment
StatefulSet
Pods
Service
Ingress
Volumes Claim
Helm
Release
Config Maps
Secret
Store
Docker Images
Orchestration Templa...
App Catalog
ack-hyperledger-fabric 1.1.0 incubator
ack-istio 1.0.2 incubator
ack-istio-remote 1.0.2 incubator
ack-openmpi 3.1.0 incubator
ack-springcloud-configserver 1.5.13.RELEASE incubator
ack-springcloud-eureka 1.5.13.RELEASE incubator
ack-springcloud-hystrix 1.5.13.RELEASE incubator
ack-springcloud-turbine 1.5.13.RELEASE incubator
ack-springcloud-zipkin 1.5.13.RELEASE incubator
ack-springcloud-zuul 1.5.13.RELEASE incubator
ack-tensorflow-dev 1.5.0 incubator
ack-tensorflow-serving 1.4.0 incubator

3. Click **ack-istio**.
4. Click the **Values** tab to configure parameters.



Helm chart of all istio components for Kubernetes on Alibaba Cloud Container Service

Only Kubernetes versions 1.8.4 and above are supported. For clusters of version 1.8.1, you can perform "upgrade cluster" operation in the cluster list

```
# tracing(jaeger on AliCloud Log Service)
tracing-on-sls:
  enabled: false
  storage:
    type: aliyun-log
    aliyun_sls:
      project: newlogsample
      logstore: mylogstore1
      endpoint: cn-hangzhou.log.aliyuncs.com
      accesskey:
        id:
        secret:

# Common settings.
global:
  # Default hub for Istio images.
  hub: registry.cn-hangzhou.aliyuncs.com/aliacs-app-catalog

  # Default tag for Istio images.
  tag: 1.0.2

  # Gateway used for legacy k8s Ingress resources. By default it is
  # using 'istio:ingress', to match 0.8 config. It requires that
  # ingress.enabled is set to true. You can also set it
  # to ingressgateway, or any other gateway you define in the 'gateway'
```

Note:

- For information about the description, values, and defaults of general parameters, see the **Configuration** section on the **Readme** tab page.
- You can also customize parameters. For example, whether to start **`grafana`**, **`prometheus`**, **`tracing`**, **`weave-scope`**, and **`kiali`**. See the following:

```
#
```

```
# addons configuration
#
grafana:
enabled: true
replicaCount: 1
image: istio-grafana
service:
name: http
type: ClusterIP
externalPort: 3000
internalPort: 3000
....
prometheus:
enabled: true
replicaCount: 1
image:
repository: registry.cn-hangzhou.aliyuncs.com/aliacs-app-catalog/
istio-prometheus
tag: latest
....
tracing:
enabled: true
jaeger:
enabled: true
....
weave-scope:
enabled: true
global:
# global.image: the image that will be used for this release
image:
repository: weaveworks/scope
tag: "1.9.0"
# global.image.pullPolicy: must be Always, IfNotPresent, or Never
pullPolicy: "IfNotPresent"
....
kiali:
enabled: true
replicaCount: 1
image:
repository: registry.cn-hangzhou.aliyuncs.com/aliacs-app-catalog/
istio-kiali
tag: dev
```

5. In the **Deploy** section on the right, configure the following.

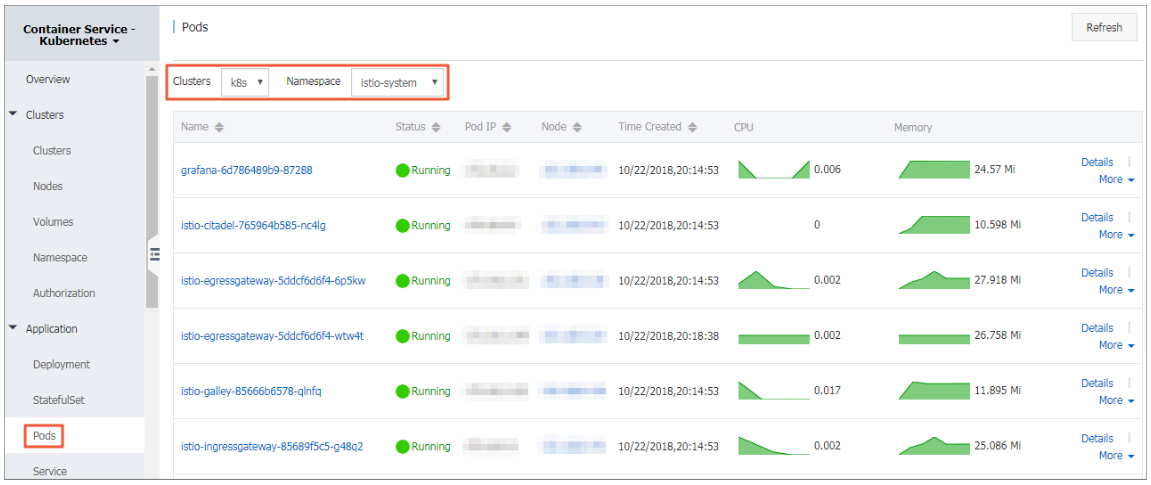
Configuration	Description
Clusters	Target cluster in which Istio is deployed.
Namespace	Namespace in which Istio is deployed. The default namespace is default.
Release Name	Name of Istio to be released.

6. Click **DEPLOY** to start deployment.

Expected results:

- • In the left-side navigation pane, click **Application** > **Pods**.

- Select the cluster and namespace in which Istio is deployed, and you can see the relevant pods in which Istio is deployed.

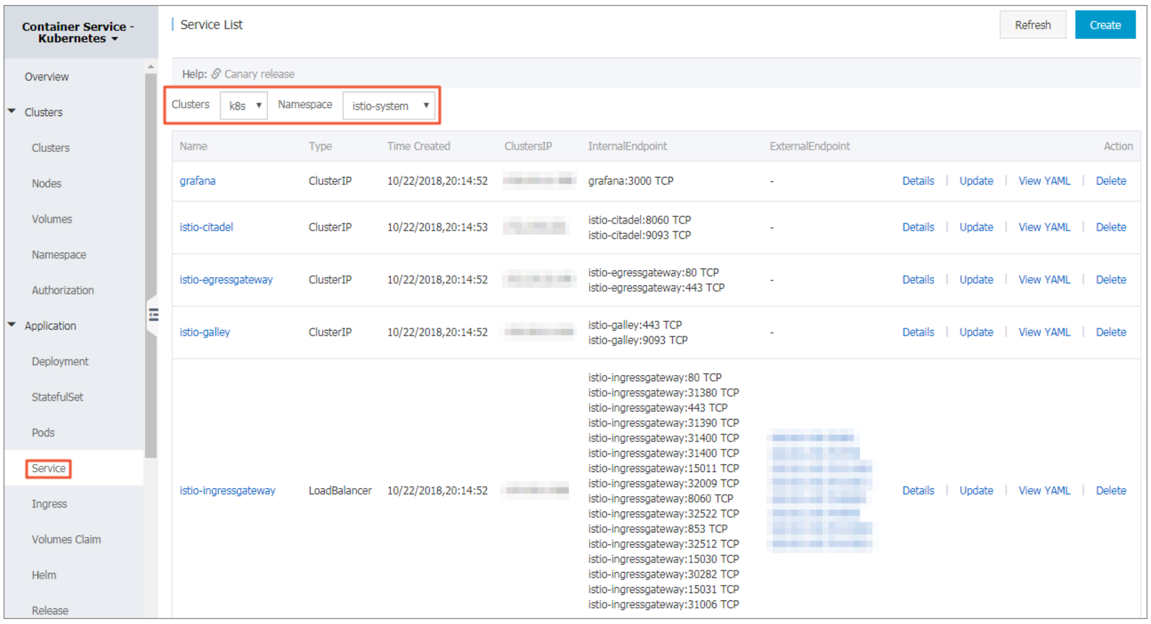


Pods

Clusters: k8s   Namespace: istio-system

Name	Status	Pod IP	Node	Time Created	CPU	Memory	
grafana-6d786489b9-87288	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0.006	24.57 Mi	Details | More
istio-citadel-765964b585-nc4lg	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0	10.598 Mi	Details | More
istio-egressgateway-5ddcf6d6f4-6p5kw	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0.002	27.918 Mi	Details | More
istio-egressgateway-5ddcf6d6f4-wtw4t	Running	[redacted]	[redacted]	10/22/2018,20:18:38	0.002	26.758 Mi	Details | More
istio-galley-85666b6578-qlnfq	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0.017	11.895 Mi	Details | More
istio-ingressgateway-85689f5c5-g48q2	Running	[redacted]	[redacted]	10/22/2018,20:14:53	0.002	25.086 Mi	Details | More

- - In the left-side navigation pane, click **Application** > **Service**.
  - Select the cluster and namespace in which Istio is deployed, and you can see the access addresses provided by the relevant services in which Istio is deployed.



Service List

Help: Canary release

Clusters: k8s   Namespace: istio-system

Name	Type	Time Created	ClustersIP	InternalEndpoint	ExternalEndpoint	Action
grafana	ClusterIP	10/22/2018,20:14:52	[redacted]	grafana:3000 TCP	-	Details | Update | View YAML | Delete
istio-citadel	ClusterIP	10/22/2018,20:14:53	[redacted]	istio-citadel:8060 TCP istio-citadel:9093 TCP	-	Details | Update | View YAML | Delete
istio-egressgateway	ClusterIP	10/22/2018,20:14:52	[redacted]	istio-egressgateway:80 TCP istio-egressgateway:443 TCP	-	Details | Update | View YAML | Delete
istio-galley	ClusterIP	10/22/2018,20:14:52	[redacted]	istio-galley:443 TCP istio-galley:9093 TCP	-	Details | Update | View YAML | Delete
istio-ingressgateway	LoadBalancer	10/22/2018,20:14:52	[redacted]	istio-ingressgateway:80 TCP istio-ingressgateway:31380 TCP istio-ingressgateway:443 TCP istio-ingressgateway:31390 TCP istio-ingressgateway:31400 TCP istio-ingressgateway:31400 TCP istio-ingressgateway:15011 TCP istio-ingressgateway:32009 TCP istio-ingressgateway:8060 TCP istio-ingressgateway:32522 TCP istio-ingressgateway:853 TCP istio-ingressgateway:32512 TCP istio-ingressgateway:15030 TCP istio-ingressgateway:30282 TCP istio-ingressgateway:15031 TCP istio-ingressgateway:31006 TCP	[redacted]	Details | Update | View YAML | Delete

1.14.3 Update Istio

You can modify the deployed Istio through **updates**.

Prerequisites

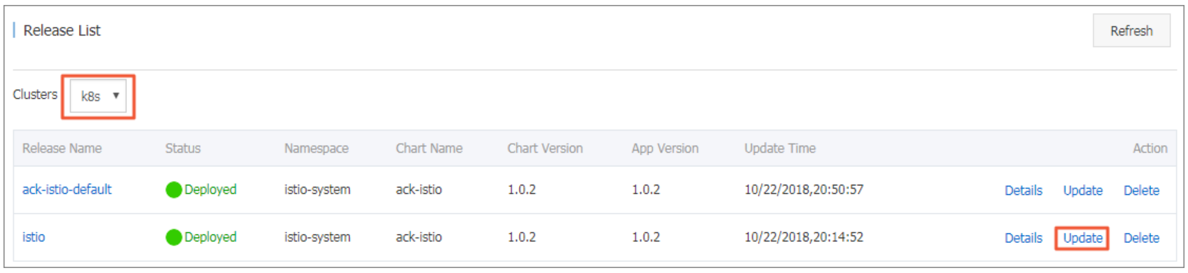
- You have created an Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created an **Istio**. For more information, see *Deploy Istio*.

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:

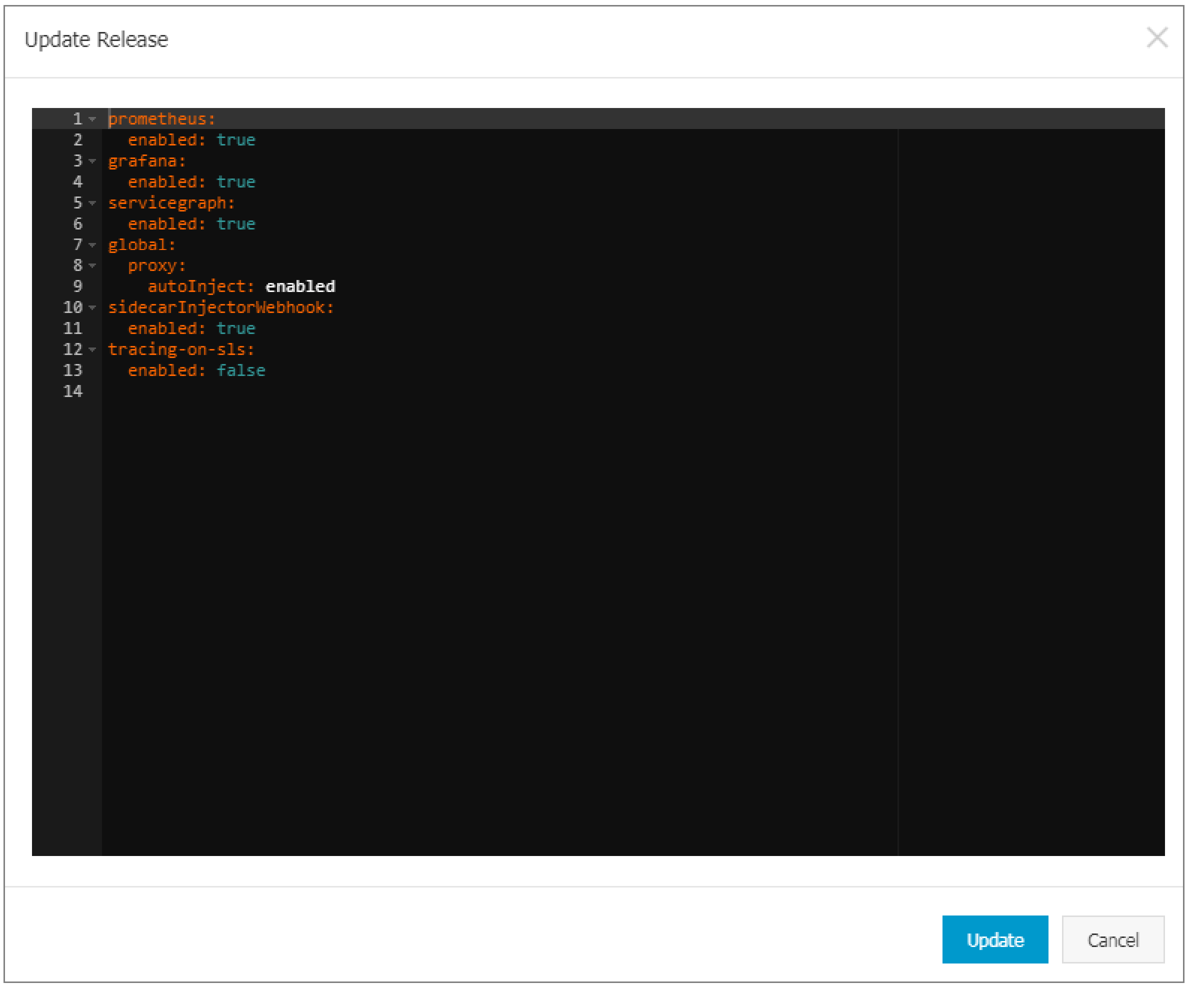
- The **release name** of the Istio that is deployed through the cluster interface is istio. Configurations to be updated are the same as the options configured in deployment.
- The **release name** of the Istio that is deployed through the application catalog is the name specified when you create the Istio. Configurations to be updated are the same as the options configured in deployment.



Release Name	Status	Namespace	Chart Name	Chart Version	App Version	Update Time	Action
ack-istio-default	Deployed	istio-system	ack-istio	1.0.2	1.0.2	10/22/2018,20:50:57	Details Update Delete
istio	Deployed	istio-system	ack-istio	1.0.2	1.0.2	10/22/2018,20:14:52	Details Update Delete

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:



```
prometheus:
  enabled: true
grafana:
  enabled: true
servicegraph:
  enabled: true
global:
  proxy:
    autoInject: enabled
sidecarInjectorWebhook:
  enabled: true
tracing-on-sls:
  enabled: false
```

Result

You can view updated content in two ways:

- After you complete the update, the page automatically jumps to the**Release 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.

1.14.4 Delete Istio

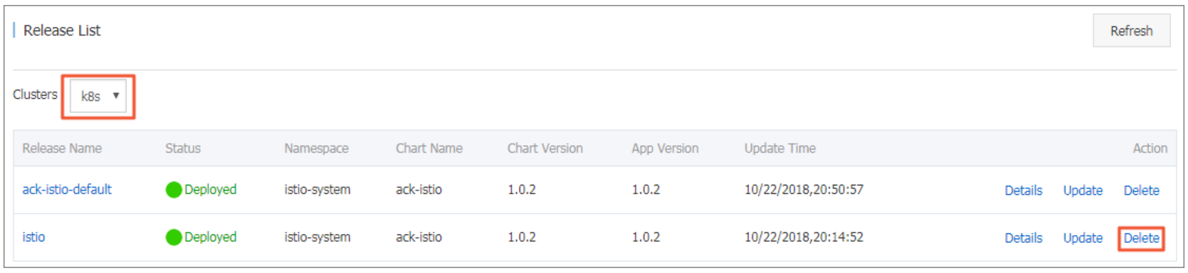
You can delete a deployed Istio through the **deleting**operation.

Prerequisites

- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created an **Istio**. For more information, see *Deploy Istio*.

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 deleted, and click **Delete** in the action column.



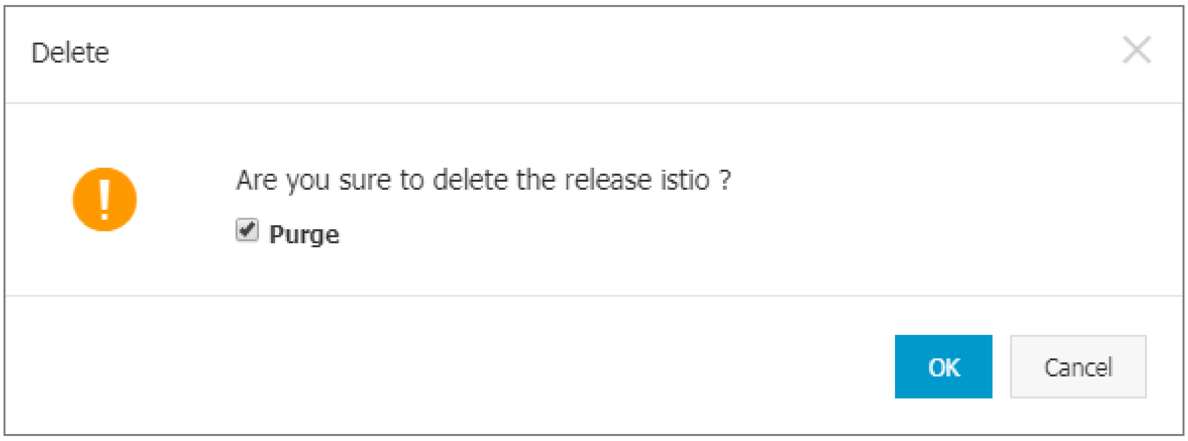
Release List

Refresh

Clusters: k8s

Release Name	Status	Namespace	Chart Name	Chart Version	App Version	Update Time	Action
ack-istio-default	● Deployed	istio-system	ack-istio	1.0.2	1.0.2	10/22/2018,20:50:57	Details Update Delete
istio	● Deployed	istio-system	ack-istio	1.0.2	1.0.2	10/22/2018,20:14:52	Details Update Delete

4. In the displayed dialog box, click **OK**.



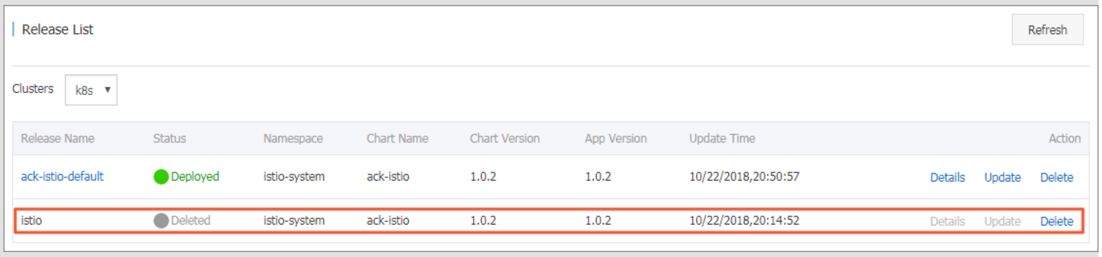
Delete

Are you sure to delete the release istio ?

[✓] Purge

Note:

- Do not select the **Purge** box:
  - Releasing records are not deleted:



Release List

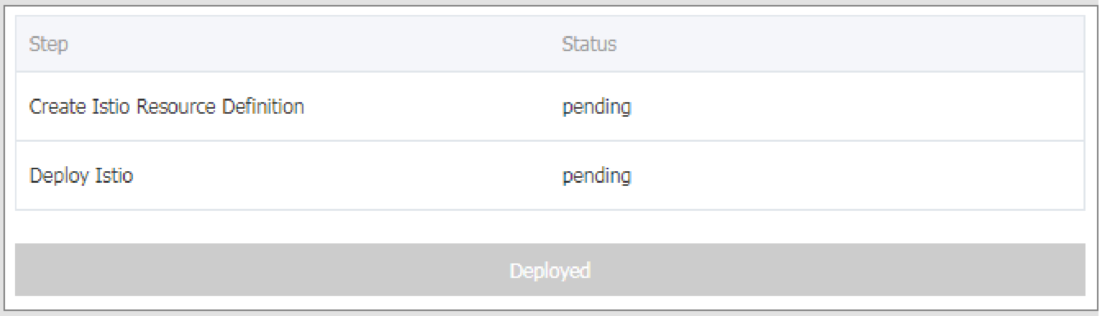
Refresh

Clusters: k8s

Release Name	Status	Namespace	Chart Name	Chart Version	App Version	Update Time	Action
ack-istio-default	● Deployed	istio-system	ack-istio	1.0.2	1.0.2	10/22/2018,20:50:57	Details Update Delete
istio	● Deleted	istio-system	ack-istio	1.0.2	1.0.2	10/22/2018,20:14:52	Details Update Delete

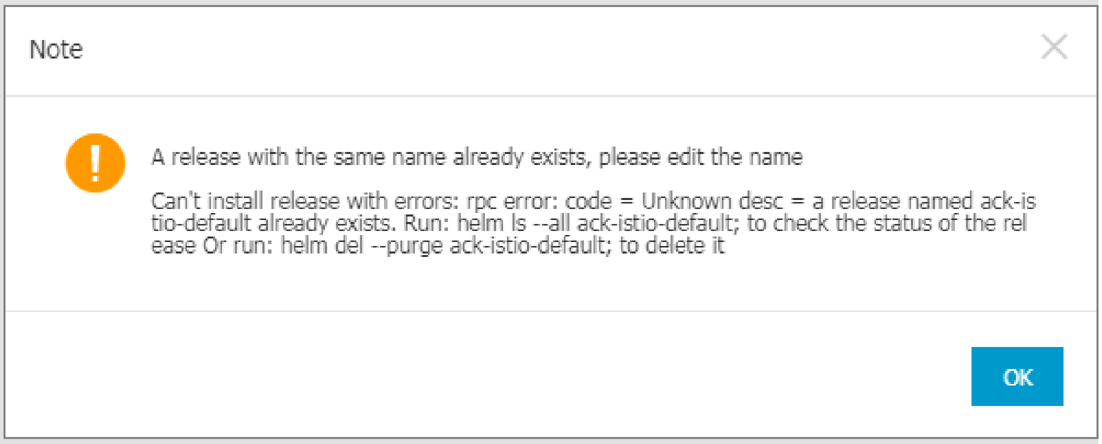
  - The name of this Istio cannot be used again.

    When you redeploy the Istio through the cluster interface, the deployment status is deployed.



Step	Status
Create Istio Resource Definition	pending
Deploy Istio	pending

When you redeploy the Istio through the application catalog, the system prompts you that deployment or resource with the same name already exists and please modify the Istio name.



Note

A release with the same name already exists, please edit the name

Can't install release with errors: rpc error: code = Unknown desc = a release named ack-istio-default already exists. Run: helm ls --all ack-istio-default; to check the status of the release Or run: helm del --purge ack-istio-default; to delete it

- Selecting the **Purge** box deletes all releasing records and the Istio name can be reused.

We recommend that you keep the **Purge** box selected.

Result

Back to the **Release List** page, you can see that the Istio is removed.

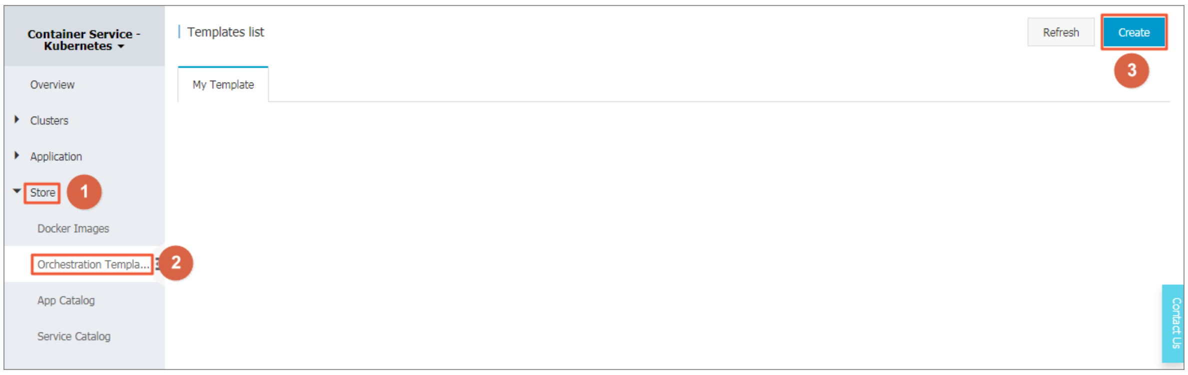
1.15 Template

1.15.1 Create an orchestration template

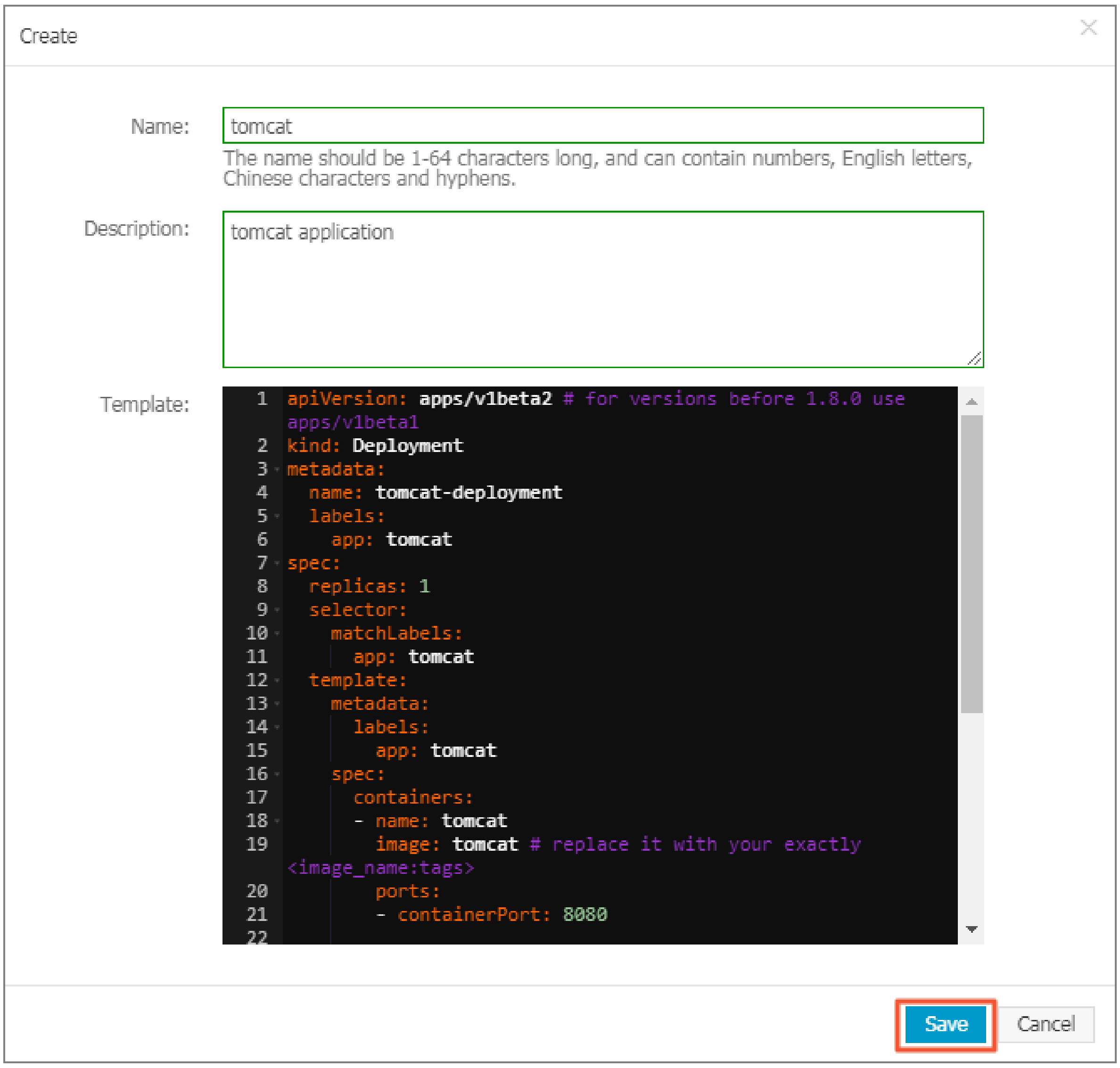
You can use multiple methods to create orchestration templates through the Container Service console.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Store** > **Orchestration Templates** in the left-side navigation pane. Click **Create** in the upper-right corner.



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 `---`.



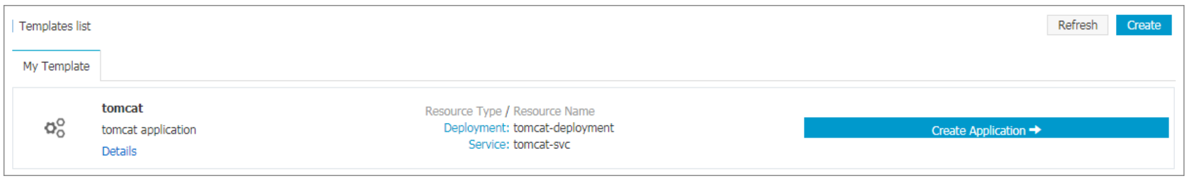
tomcat

The name should be 1-64 characters long, and can contain numbers, English letters, Chinese characters and hyphens.

tomcat application

```yaml
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment
  labels:
    app: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat # replace it with your exactly <image_name:tags>
        ports:
        - containerPort: 8080
```

4. After the template is created, the **Template List** page is displayed. You can see the template under **My template**.

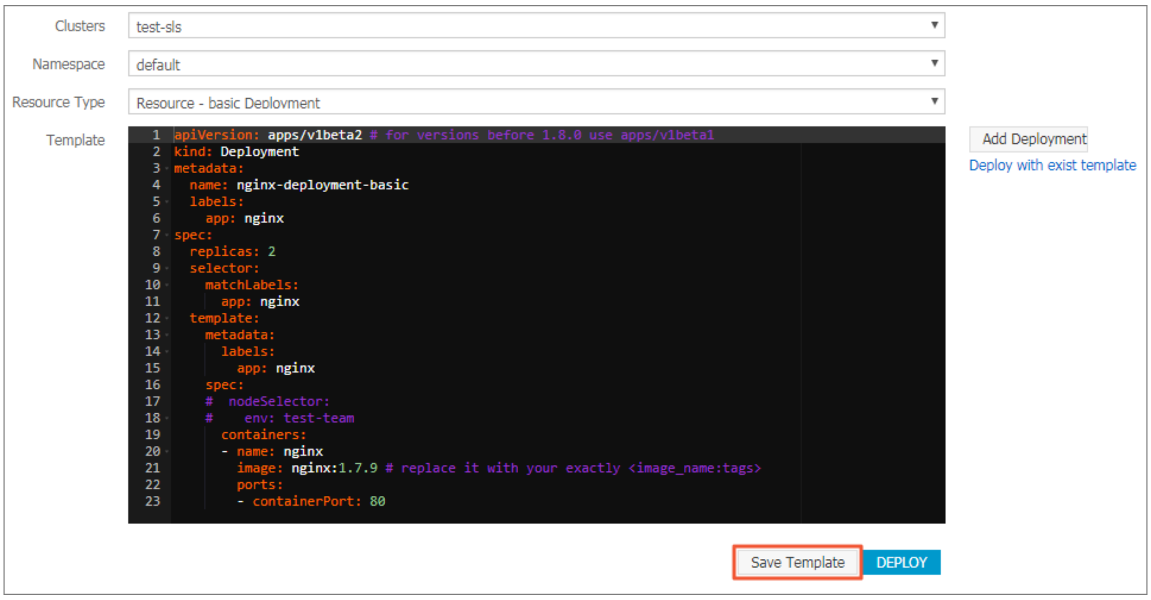


tomcat
tomcat application

Resource Type / Resource Name
Deployment: tomcat-deployment
Service: tomcat-svc

5. Optional: You can also click **Application** > **Deployment** in the left-hand navigation pane, and click **Create by template** to enter the **Deploy templates** page. Save one of orchestration templates built-in Container Service as your custom template.

   a) Select a built-in template and click **Save Template**.



Clusters: test-sls

Namespace: default

Resource Type: Resource - basic Deployment

Template:

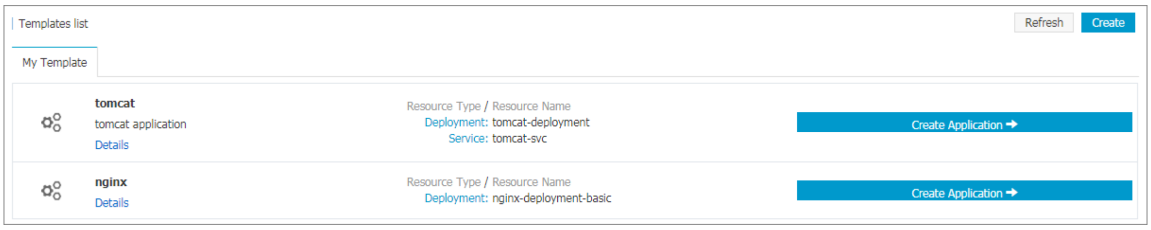
```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment-basic
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
    #  nodeSelector:
    #    env: test-team
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
        ports:
        - containerPort: 80
```

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) Click **Store** > **Orchestration Template**, the created template is displayed under **My Template**.



Templates list

My Template

tomcat tomcat application Details	Resource Type / Resource Name Deployment: tomcat-deployment Service: tomcat-svc	Create Application ➔
nginx Details	Resource Type / Resource Name Deployment: nginx-deployment-basic	Create Application ➔

What's next

You can quickly create an application by using the orchestration template under **My Template**.

1.15.2 Edit an orchestration template

You can edit an orchestration arrangement template.

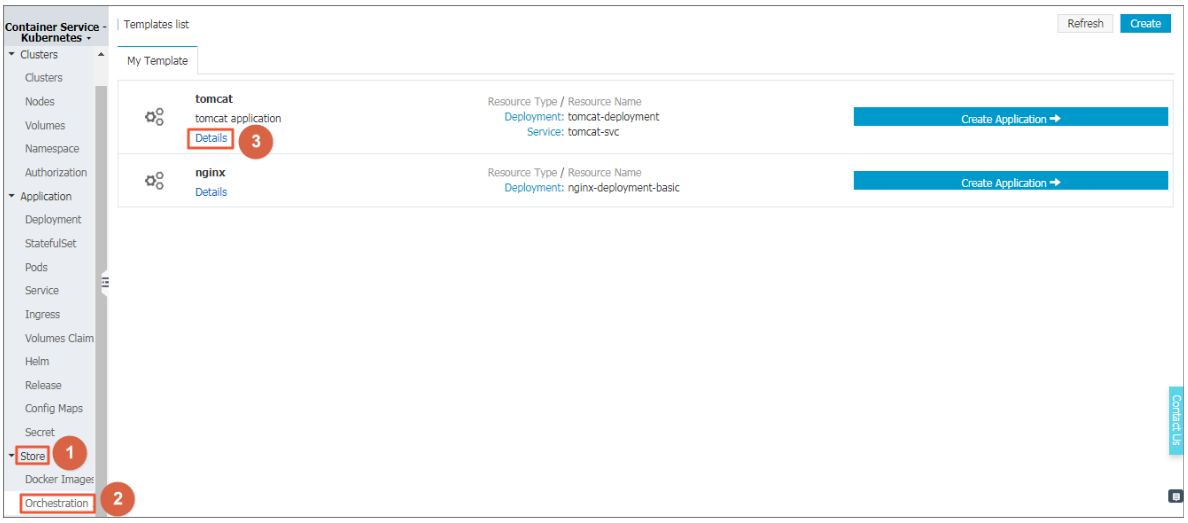
Prerequisites

You have created an orchestration template, see *Create an orchestration template*.

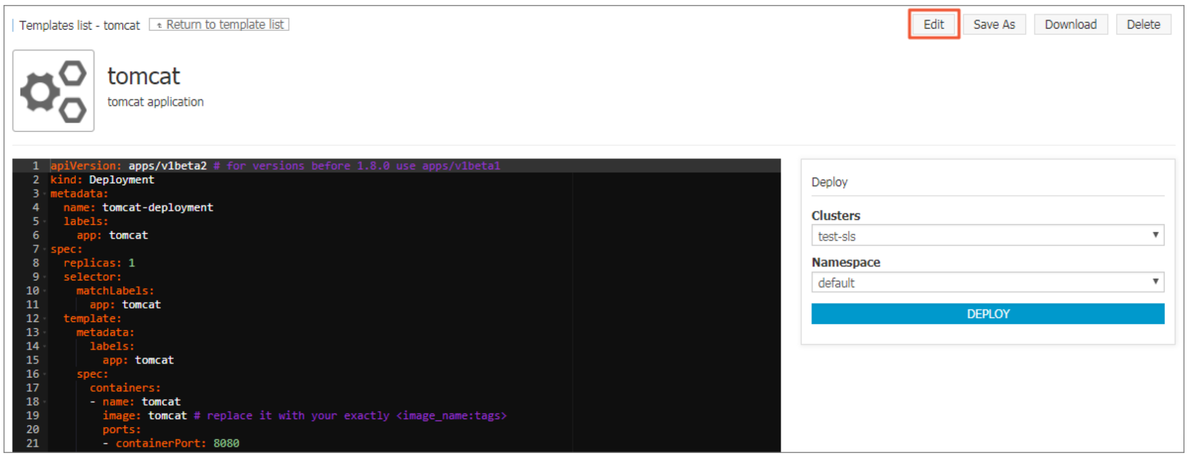
Procedure

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**.

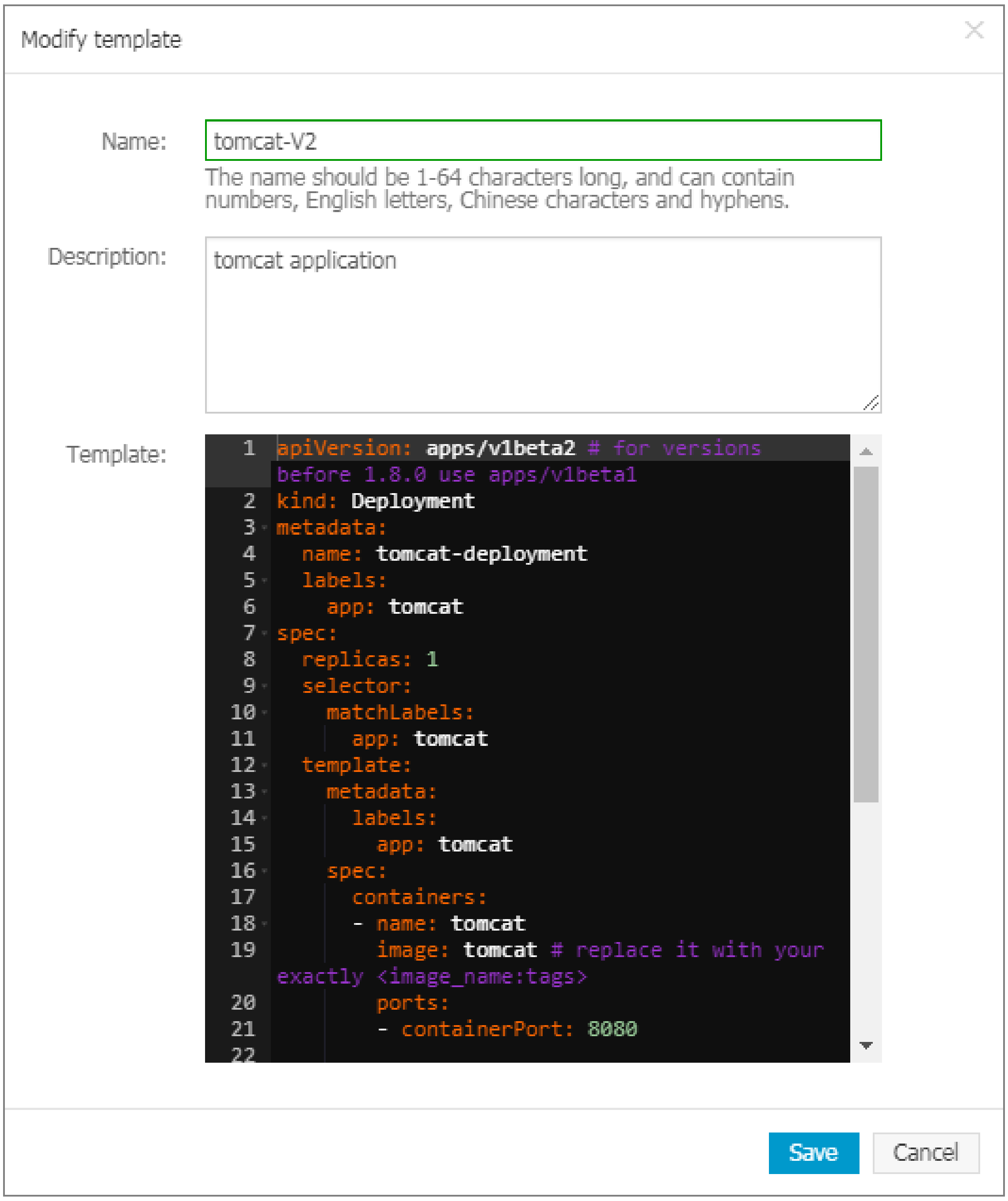


4. Click **Edit** in the upper-right corner.



```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment
  labels:
    app: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat # replace it with your exactly <image_name:tags>
        ports:
        - containerPort: 8080
```

5. In the displayed dialog box, edit the name, description, and template, and click **Save**.



Name: tomcat-V2

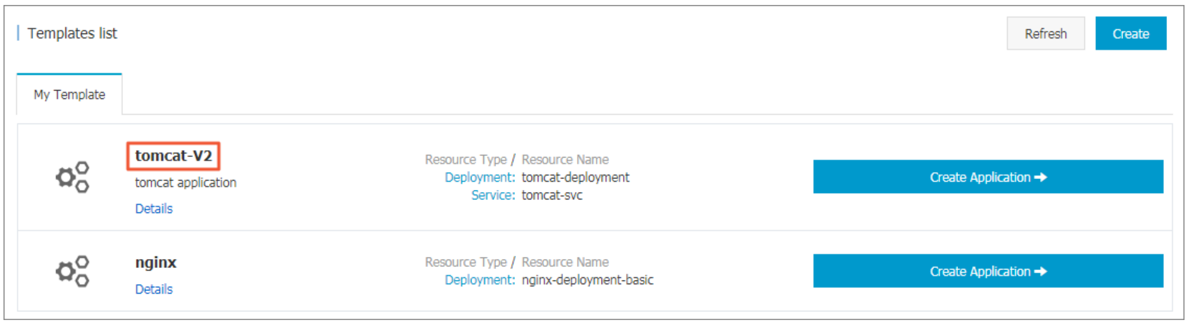
The name should be 1-64 characters long, and can contain numbers, English letters, Chinese characters and hyphens.

Description: tomcat application

Template:

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment
  labels:
    app: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat # replace it with your exactly <image_name:tags>
        ports:
        - containerPort: 8080
```

6. Back to the **Template List** page, under **My Template**, you can see the template is changed.



tomcat-V2
tomcat application
Resource Type / Resource Name
Deployment: tomcat-deployment
Service: tomcat-svc

nginx
Resource Type / Resource Name
Deployment: nginx-deployment-basic

1.15.3 Save an existing orchestration template as a new one

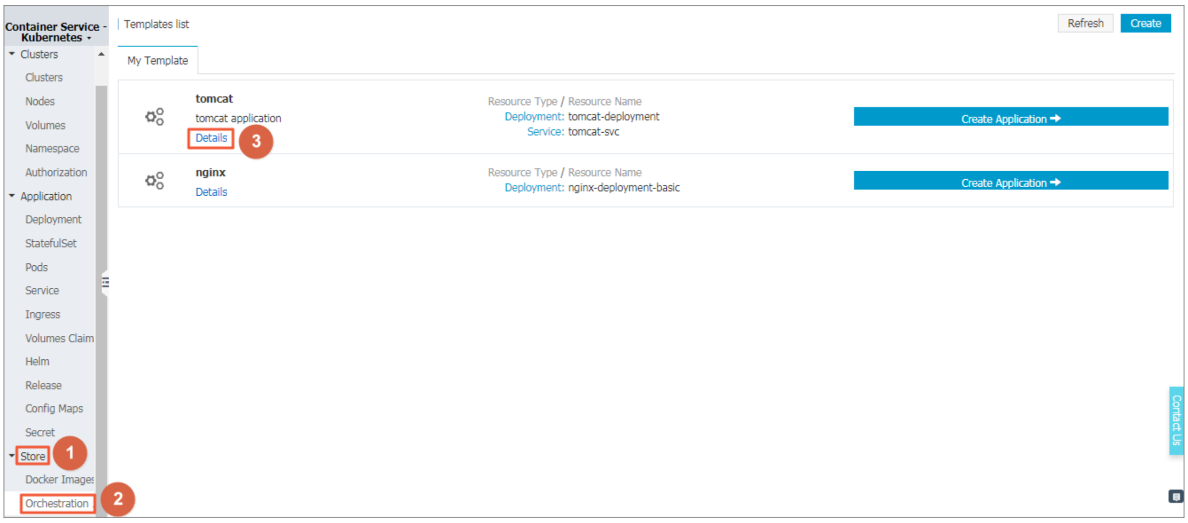
You can save an existing template as a new one.

Prerequisites

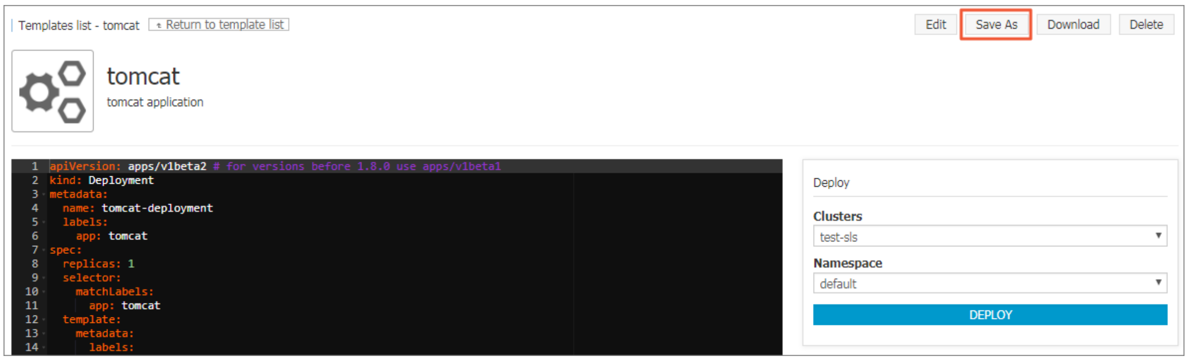
You have created an orchestration template, see *Create an orchestration template*.

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**.

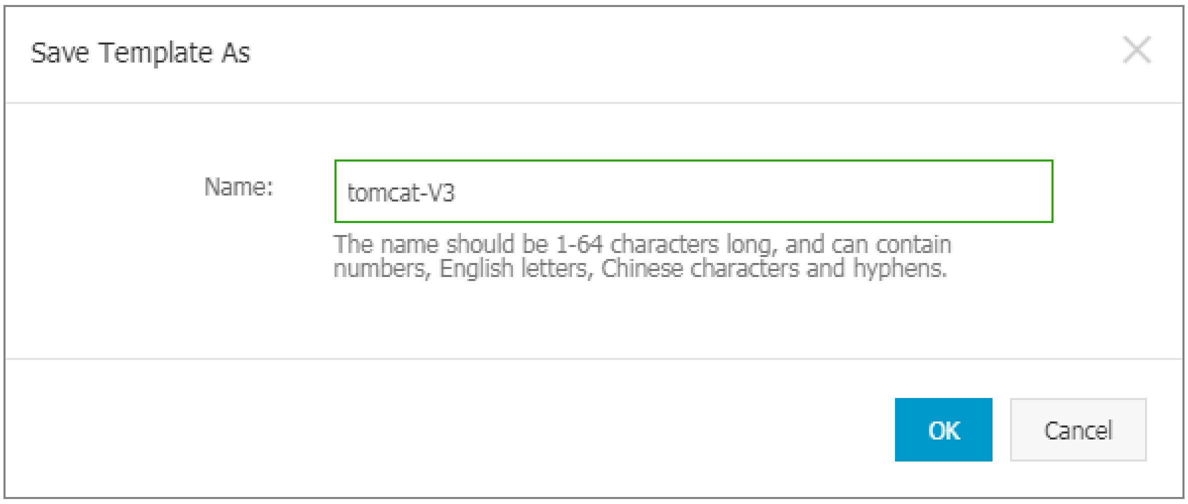


4. You can modify the template and click **Save as** in the upper-right corner.



```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment
  labels:
    app: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
```

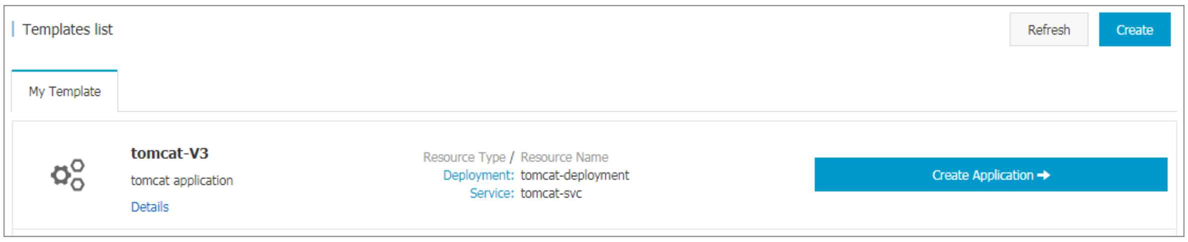
5. In the displayed dialog box, configure the template name and click **OK**.


Save Template As

Name: tomcat-V3

The name should be 1-64 characters long, and can contain numbers, English letters, Chinese characters and hyphens.

6. Back to the **Template List** page, you can see that the saved template is displayed under **My Template**.


Templates list

My Template

tomcat-V3

tomcat application

Resource Type / Resource Name

Deployment: tomcat-deployment

Service: tomcat-svc

1.15.4 Download an orchestration template

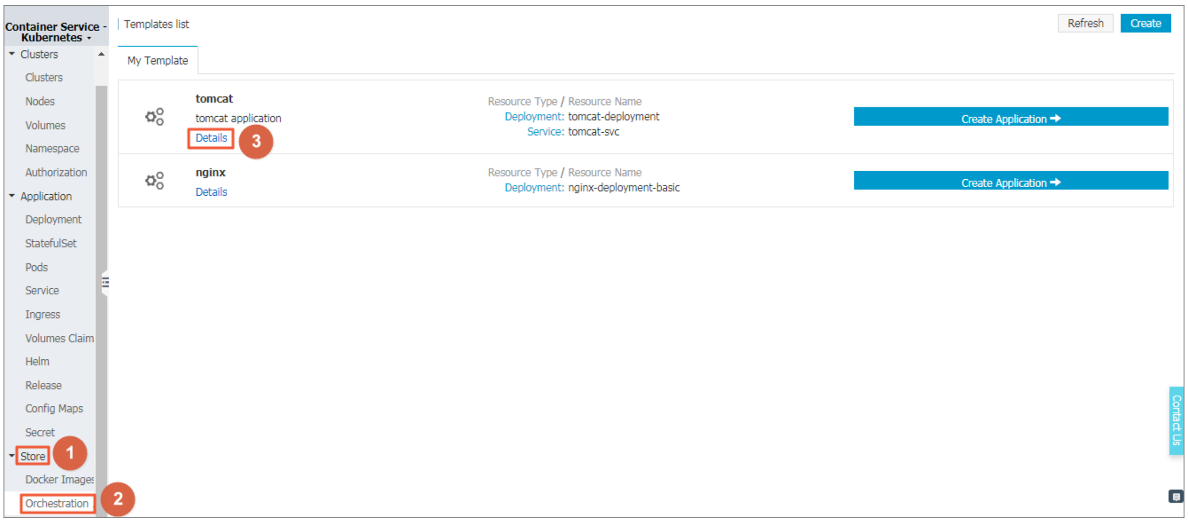
You can download an existing orchestration template.

Prerequisites

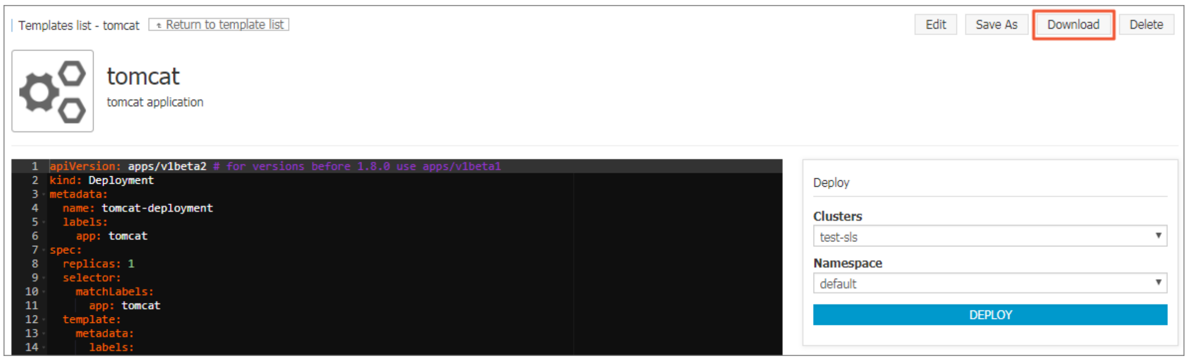
You have created an orchestration template, see *Create an orchestration template*.

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.



```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment
  labels:
    app: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
```

1.15.5 Delete an orchestration template

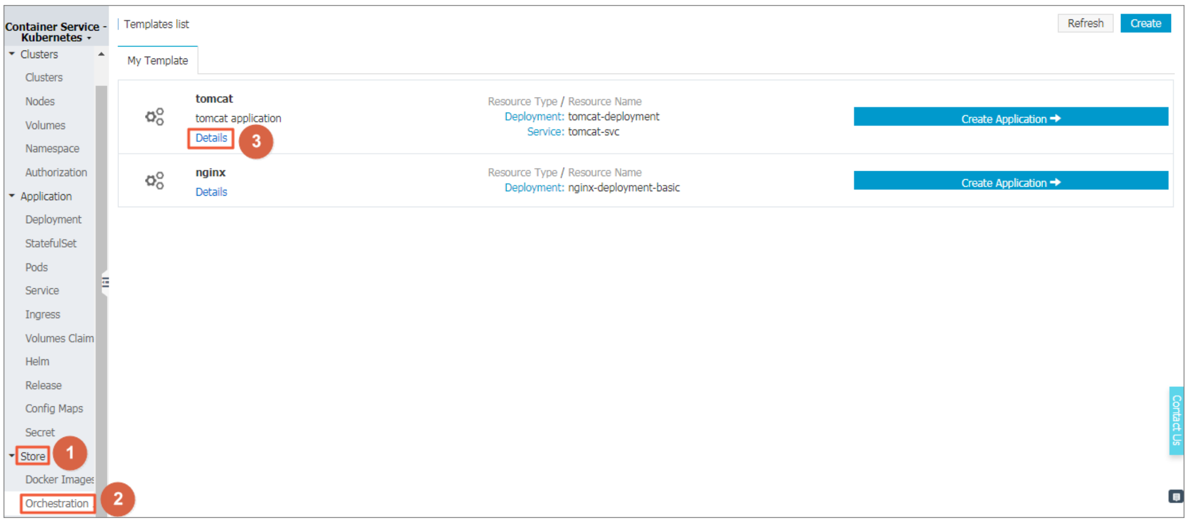
You can delete an orchestration template that is no longer needed.

Prerequisites

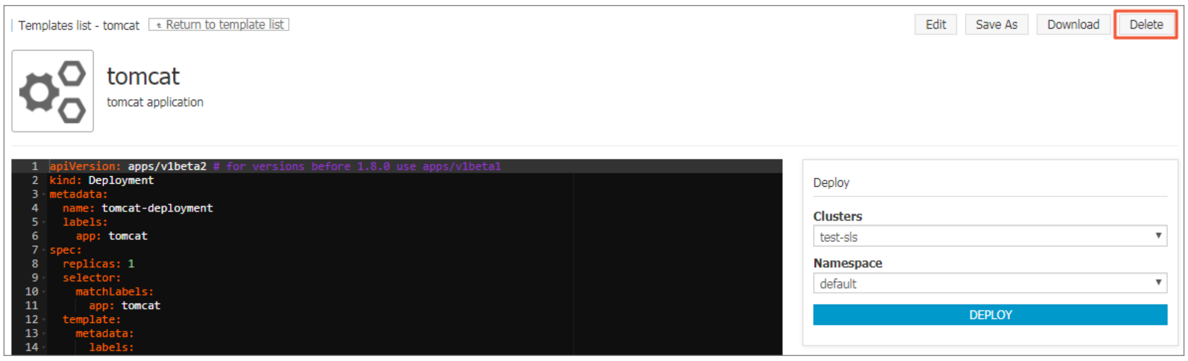
You have created an orchestration template, see *Create an orchestration template*.

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.



```yaml
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: tomcat-deployment
  labels:
    app: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
```

5. Click **Confirm** in the displayed dialog box.

1.16 App catalog

1.16.1 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 application into several microservi ces, 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

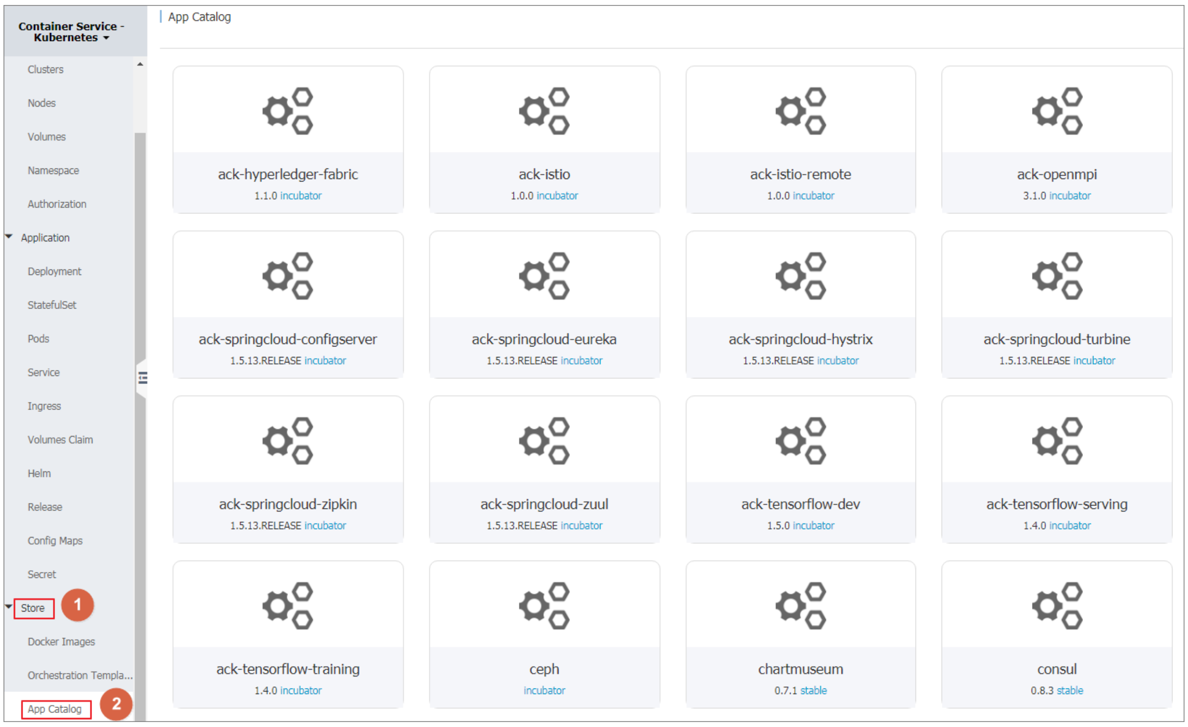
Currently, you can deploy and manage the charts in the app catalog by using the Helm tool. For more information, see *Simplify Kubernetes application deployment by using Helm*.

1.16.2 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.


Container Service - Kubernetes
Clusters
Nodes
Volumes
Namespace
Authorization
Application
Deployment
StatefulSet
Pods
Service
Ingress
Volumes Claim
Helm
Release
Config Maps
Secret
Store
1
Docker Images
Orchestration Templa...
App Catalog
2
App Catalog
ack-hyperledger-fabric 1.1.0 incubator
ack-istio 1.0.0 incubator
ack-istio-remote 1.0.0 incubator
ack-openmpi 3.1.0 incubator
ack-springcloud-configserver 1.5.13.RELEASE incubator
ack-springcloud-eureka 1.5.13.RELEASE incubator
ack-springcloud-hystrix 1.5.13.RELEASE incubator
ack-springcloud-turbine 1.5.13.RELEASE incubator
ack-springcloud-zipkin 1.5.13.RELEASE incubator
ack-springcloud-zuul 1.5.13.RELEASE incubator
ack-tensorflow-dev 1.5.0 incubator
ack-tensorflow-serving 1.4.0 incubator
ack-tensorflow-training 1.4.0 incubator
ceph incubator
chartmuseum 0.7.1 stable
consul 0.8.3 stable

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*.

1.17 Service catalog

1.17.1 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 applicatio ns 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 applicatio ns.

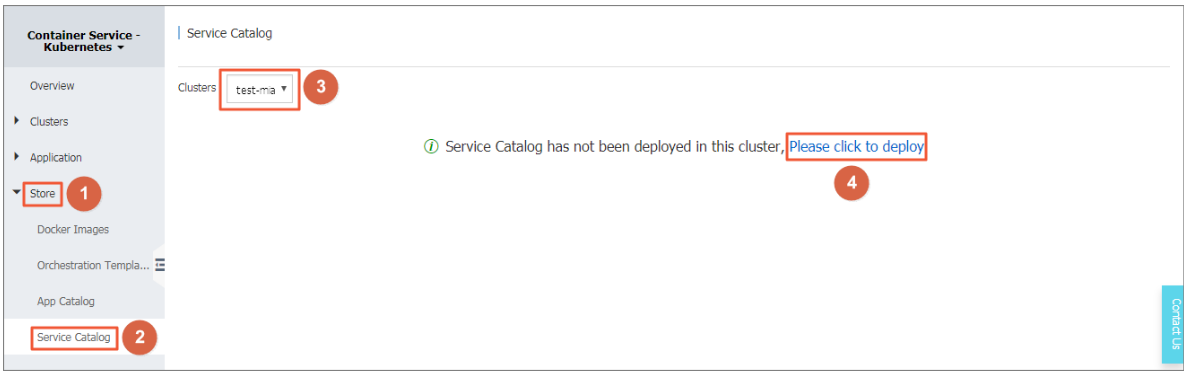
Alibaba Cloud Container Service supports and integrates with the service catalog function. The service catalog function aims to access and manage the service brokers, which allows applicatio ns running in Kubernetes clusters to use the managed services offered by service brokers. A series of infrastructure 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*.

1.17.2 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.

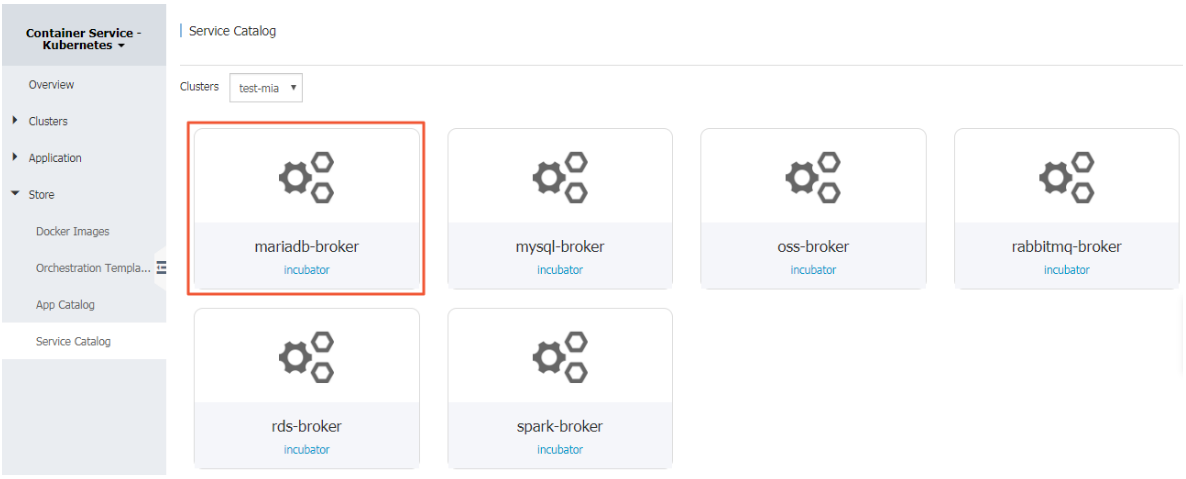


Service Catalog has not been deployed in this cluster,

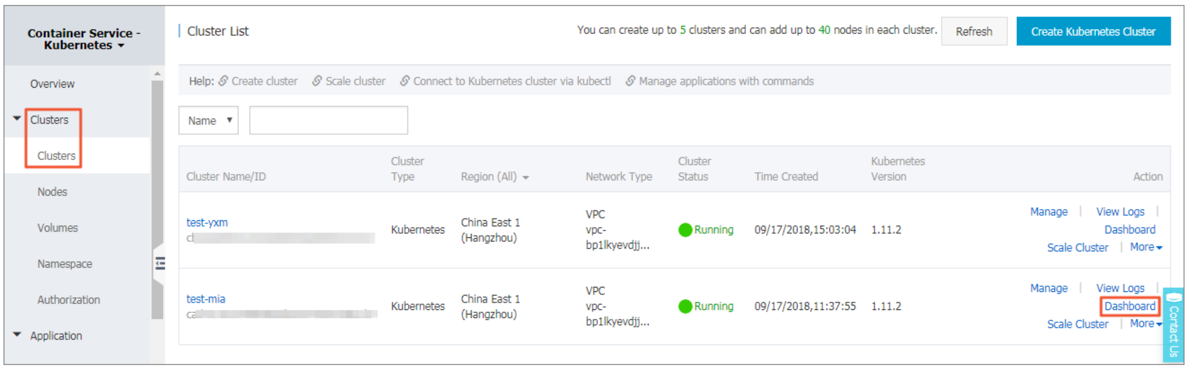
4. After the installation, the service broker, which is installed by default, is displayed on the Service Catalog page. You can click the mariadb-broker to view the details.

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.



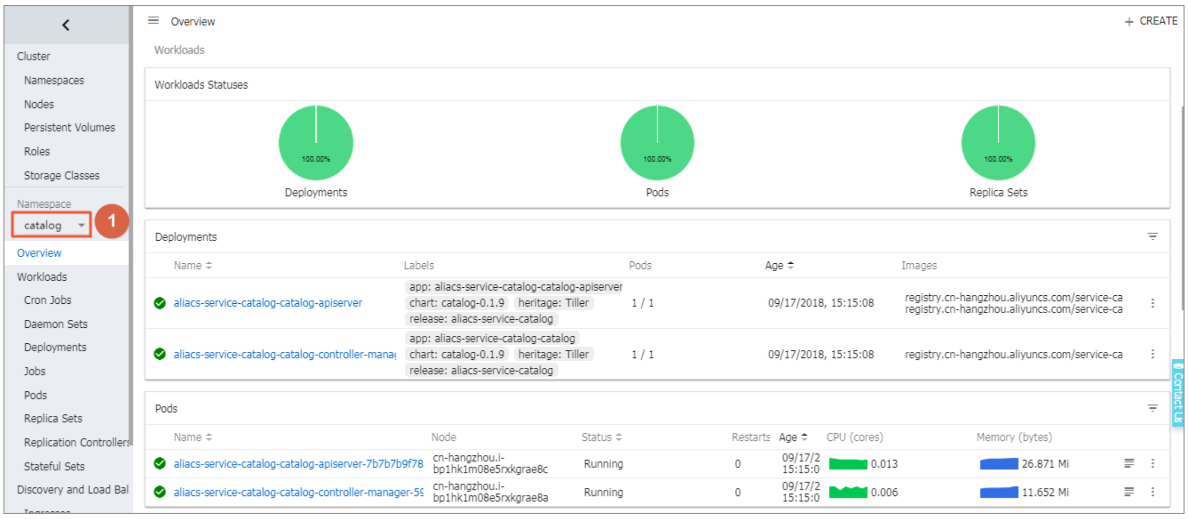
5. Click **Clusters** in the left-side navigation pane. Click **Dashboard** at the right of a cluster.



You can create up to 5 clusters and can add up to 40 nodes in each cluster.

Cluster Name/ID	Cluster Type	Region (All)	Network Type	Cluster Status	Time Created	Kubernetes Version	Action
test-yxm [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1lkyevdjj...	Running	09/17/2018,15:03:04	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More
test-mia [redacted]	Kubernetes	China East 1 (Hangzhou)	VPC vpc-bp1lkyevdjj...	Running	09/17/2018,11:37:55	1.11.2	Manage | View Logs | Dashboard Scale Cluster | More

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.



Deployments

	Name	Labels	Pods	Age	Images
✓	aliacs-service-catalog-catalog-apiserver	app: aliacs-service-catalog-catalog-apiserver chart: catalog-0.1.9 heritage: Tiller release: aliacs-service-catalog	1 / 1	09/17/2018, 15:15:08	registry.cn-hangzhou.aliyuncs.com/service-ca registry.cn-hangzhou.aliyuncs.com/service-ca
✓	aliacs-service-catalog-catalog-controller-mana	app: aliacs-service-catalog-catalog chart: catalog-0.1.9 heritage: Tiller release: aliacs-service-catalog	1 / 1	09/17/2018, 15:15:08	registry.cn-hangzhou.aliyuncs.com/service-ca

Pods

	Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
✓	aliacs-service-catalog-catalog-apiserver-7b7b7b9f78	cn-hangzhou.i-bp1hk1m08e5rxkgrae8c	Running	0	09/17/2 15:15:0	0.013	26.871 Mi
✓	aliacs-service-catalog-catalog-controller-manager-5	cn-hangzhou.i-bp1hk1m08e5rxkgrae8a	Running	0	09/17/2 15:15:0	0.006	11.652 Mi

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.