Alibaba Cloud
Aliyun Container for Kubernetes

User Guide

Issue: 20190217

Legal disclaimer

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

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified , reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates . The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning informatio n, supplementary instructions, and other content that the user must understand.	Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click **OK**.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *`[-all|-t]`*

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

Contents

1 Kubernetes cluster

1.1 Security bulletins

1.1.1 Vulnerability fix: `CVE-2019-5736` in runc

The security vulnerability `CVE-2019-5736` in runc has been fixed for Alibaba Cloud Container Service for Kubernetes. This topic describes the impacts of this vulnerability and how to remove it.

Background

The security vulnerability may occur with Docker, containerd, or any other containers that use runc. This vulnerability gives attackers the ability to use a specific container image or run the exec command to obtain the file handle used by the running host runc. Attackers can overwrite the host runc binary file, then obtain root permission to access the host, and execute commands as with root permission.

For more information about security vulnerability `CVE-2019-5736`, see https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-5736.

Affected clusters

- Alibaba Cloud Container Service clusters affected by the vulnerability:
  - All Docker Swarm clusters from versions earlier than Docker v18.09.02.
  - All Kubernetes clusters except for Serverless Kubernetes clusters.
- Self-built Docker and Kubernetes clusters affected by the vulnerability:
  - All clusters that use Docker versions earlier than v18.09.2.
  - All clusters that use runc v1.0-rc6 or earlier.

> Note:
> For both of the preceding vulnerability cases, we recommend that you consult your Docker or runc vendor for a solution.

Resolution

- Method 1: Create new Kubernetes clusters of v1.11 or v1.12. Kubernetes clusters of these two versions run the latest version of Docker, which are protected from this vulnerability.
- Method 2: Upgrade the Docker version of all existing clusters to v18.09.2 or later. Using this method will interrupt your cluster containers and services.
- Method 3: Only upgrade runc. This method is applicable to clusters running Docker v17.06. We recommend that you upgrade the runc binary file of each cluster node individually to avoid a service interruption caused by upgrading the Docker engine. To upgrade a runc binary file, complete the following steps:

  1. Run the following command to locate docker-runc:

     Note:
     Usually, docker-runc is located in `/usr/bin/docker-runc`.

     ```
     which docker-runc
     ```
  2. Run the following command to back up the original runc:

     ```
     mv /usr/bin/docker-runc /usr/bin/docker-runc.orig.$(date -Iseconds
     )
     ```
  3. Run the following command to download the fixed runc:

     ```
     curl -o /usr/bin/docker-runc -sSL https://acs-public-mirror.oss-cn
     -hangzhou.aliyuncs.com/runc/docker-runc-17.06-amd64
     ```
  4. Run the following command to set permission availability for docker-runc:

     ```
     chmod +x /usr/bin/docker-runc
     ```
  5. Run the following command to test whether runc works normally:

     ```
     docker-runc -v
     # runc version 1.0.0-rc3
     # commit: fc48a25bde6fb041aae0977111ad8141ff396438
     # spec: 1.0.0-rc5
     docker run -it --rm ubuntu echo OK
     ```
  6. To upgrade the runc binary file of a Kubernetes cluster GPU node, you must also install nvidia-runtime by completing the following steps:

     a. Run the following command to locate nvidia-container-runtime:

        Note:

Usually, nvidia-container-runtime is located in */usr/bin/nvidia-container-runtime*.

```
which nvidia-container-runtime
```

b. Run the following command to back up the original nvidia-container-runtime:

```
mv /usr/bin/nvidia-container-runtime /usr/bin/nvidia-container-runtime.orig.$(date -Iseconds)
```

c. Run the following command to download the fixed nvidia-container-runtime:

```
curl -o /usr/bin/nvidia-container-runtime -sSL https://acs-public-mirror.oss-cn-hangzhou.aliyuncs.com/runc/nvidia-container-runtime-17.06-amd64
```

d. Run the following command to set permission availability for nvidia-container-runtime:

```
chmod +x /usr/bin/nvidia-container-runtime
```

e. Run the following command to test whether nvidia-container-runtime works normally:

```
nvidia-container-runtime -v
#  runc version 1.0.0-rc3
#  commit: fc48a25bde6fb041aae0977111ad8141ff396438-dirty
#  spec: 1.0.0-rc5

docker run -it --rm -e NVIDIA_VISIBLE_DEVICES=all ubuntu nvidia-smi -L
#  GPU 0: Tesla P100-PCIE-16GB (UUID: GPU-122e199c-9aa6-5063-0fd2-da009017e6dc)
```

Note:

This test is performed on a node of the GPU P100 model. Test outputs vary by GPU models.

1.1.2 Vulnerability fix: *CVE-2018-18264* for Kubernetes dashboard

Alibaba Cloud Container Service for Kubernetes has fixed dashboard vulnerability *CVE-2018-18264*. This topic describes the dashboard versions affected by the vulnerability and how to fix the vulnerability. The Kubernetes dashboards that are built in Alibaba Cloud Container Service for Kubernetes are not affected by this vulnerability because they work in the hosted form and their security settings were upgraded before the vulnerability occurred.

Background information

A security vulnerability, that is, `CVE-2018-18264`, was discovered in Kubernetes dashboards of V1.10 and earlier versions. This vulnerability allowed attackers to bypass identity authentication and read secrets within the cluster by using the dashboard logon account.

The Kubernetes dashboards that are built in Alibaba Cloud Container Service for Kubernetes are not affected by this vulnerability because they work in the hosted form and their security settings were upgraded before the vulnerability occurred.

For more information about security vulnerability `CVE-2018-18264`, see:

- *https://github.com/kubernetes/dashboard/pull/3289*
- *https://github.com/kubernetes/dashboard/pull/3400*
- *https://github.com/kubernetes/dashboard/releases/tag/v1.10.1*

Conditions required to determine that a Kubernetes dashboard is vulnerable

Your dashboard is vulnerable if you have independently deployed Kubernetes dashboard V1.10 or earlier versions (V1.7.0 to V1.10.0) that supports the logon function in your Kubernetes cluster, and you have used custom certificates.

Resolution

- If you do not need a dashboard that is deployed independently, run the following command to remove the Kubernetes dashboard from your cluster:

  ```
  kubectl --namespace kube-system delete deployment kubernetes-
  dashboard
  ```
- If you need an independently deployed dashboard, upgrade your dashboard to V1.10.1. For more information, see *https://github.com/kubernetes/dashboard/releases/tag/v1.10.1*.
- If you use the dashboard hosted by Alibaba Cloud Container Service for Kubernetes, you can continue to use your dashboard in the Container Service console because the dashboard was upgraded before the vulnerability occurred.

1.1.3 Vulnerability fix: *CVE-2018-1002105*

Alibaba Cloud has fixed system vulnerability *CVE-2018-1002105*. This topic describes the impacts of this vulnerability and how to remove it.

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

Background information

Engineers of the Kubernetes community have found security vulnerability *CVE-2018-1002105*. Kubernetes users can gain access to the backend service by forging the request and escalating the permission on the established API Server connection. Alibaba Cloud has fixed this vulnerability. To remove the vulnerability, you need to log on to the Container Service console and upgrade Kubernetes to the latest version.

For more information about the vulnerability *CVE-2018-1002105*, see *https://github.com/kubernetes/kubernetes/issues/71411*.

Affected Kubernetes versions:

- Kubernetes v1.0.x-1.9.x
- Kubernetes v1.10.0-1.10.10 (fixed in v1.10.11)
- Kubernetes v1.11.0-1.11.4 (fixed in v1.11.5)
- Kubernetes v1.12.0-1.12.2 (fixed in v1.12.3)

Affected configurations:

- Kubernetes cluster, which runs on Container Service and uses an extension API server. Furthermore, the extension API server network is directly accessible to the cluster component, kube-apiserver.
- Kubernetes cluster, which runs on Container Service and has opened permission s to interfaces such as pod exec, attach, and portforward. Then, users can use the vulnerability to obtain permissions to access all kubelet APIs of the cluster.

Cluster configuration of Alibaba Cloud Container Service for Kubernetes

- The API server of a Kubernetes cluster that runs on Container Service has RBAC enabled by default. That is, the API server denies anonymous user access through primary account authorization. Furthermore, the starting parameter of Kubelet is `anonymous-auth=false`, providing security access control against external attacks.

- If your Kubernetes cluster has multiple RAM users, the RAM users may gain unauthorized access to the backend service through interfaces such as pod exec , attach, and portforward. If your cluster has no RAM users, you do not need to worry about the vulnerability.
- RAM users do not have access to aggregate API resources by default without custom authorization from the primary account.

Solution

Log on to the Container Service console to upgrade your cluster. For more information, see *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.

1.2 Introduction

1.2.1 Overview

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

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

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

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

Limits

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

Related open-source projects

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

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

1.2.2 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 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.

Function	Description
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 capabilities. Supports upgrading a Kubernetes cluster to the latest version with one click. · Alibaba Cloud container team is responsible for solving problems about containers in your environment.

Costs and risks of self-built Kubernetes

- Building clusters is complicated

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

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

1.3 Authorization management

1.3.1 Role authorization

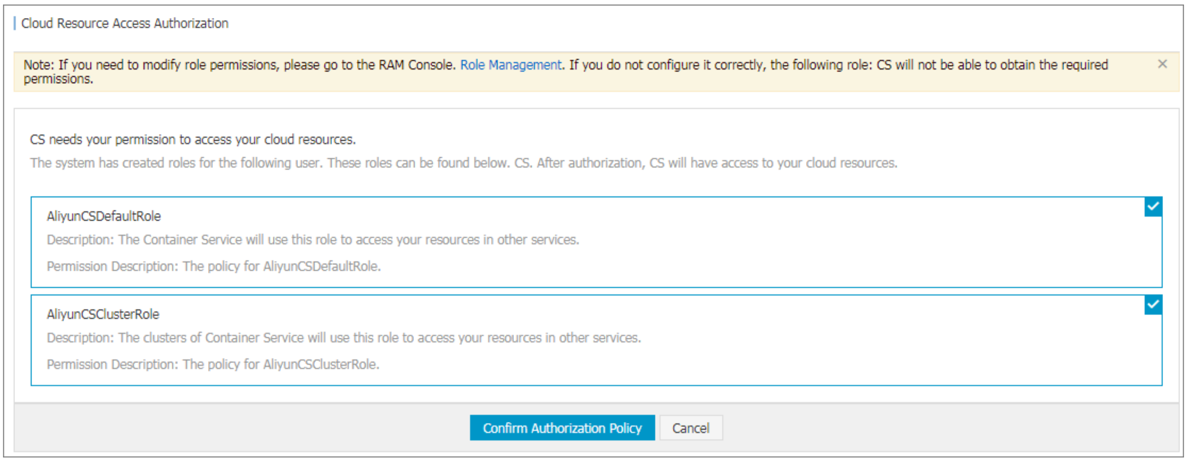
Grant the system default roles AliyunCSDefaultRole and AliyunCSClusterRole to the service account when you activate Container Service. Only after the roles are correctly granted, Container Service can normally call services such as Elastic Compute Service (ECS), Object Storage Service (OSS), Network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

Instructions

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

Procedure

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



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

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

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

Default role permissions

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

AliyunCSDefaultRole permissions

The default role AliyunCSDefaultRole contains the following main permissions:

- ECS-related permissions

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

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

- Virtual Private Cloud (VPC)-related permissions

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

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

- SLB-related permissions

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

AliyunCSClusterRole permissions

The default role AliyunCSClusterRole contains the following main permissions:

- OSS-related permissions

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

- NAS-related permissions

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

- SLB-related permissions

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

1.3.2 Use the Container Service console as a RAM user

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

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

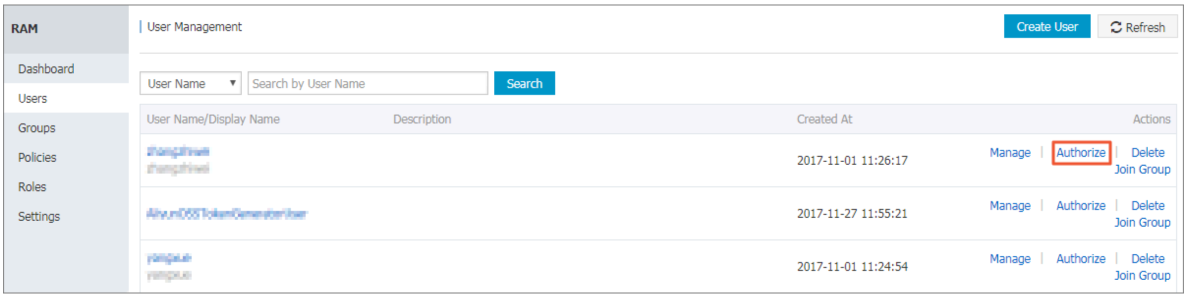
Step 1: Create a RAM user and enable console logon

1. Log on to the *RAM console*.
2. In the left-side navigation bar, click Users. Then, click Create User.

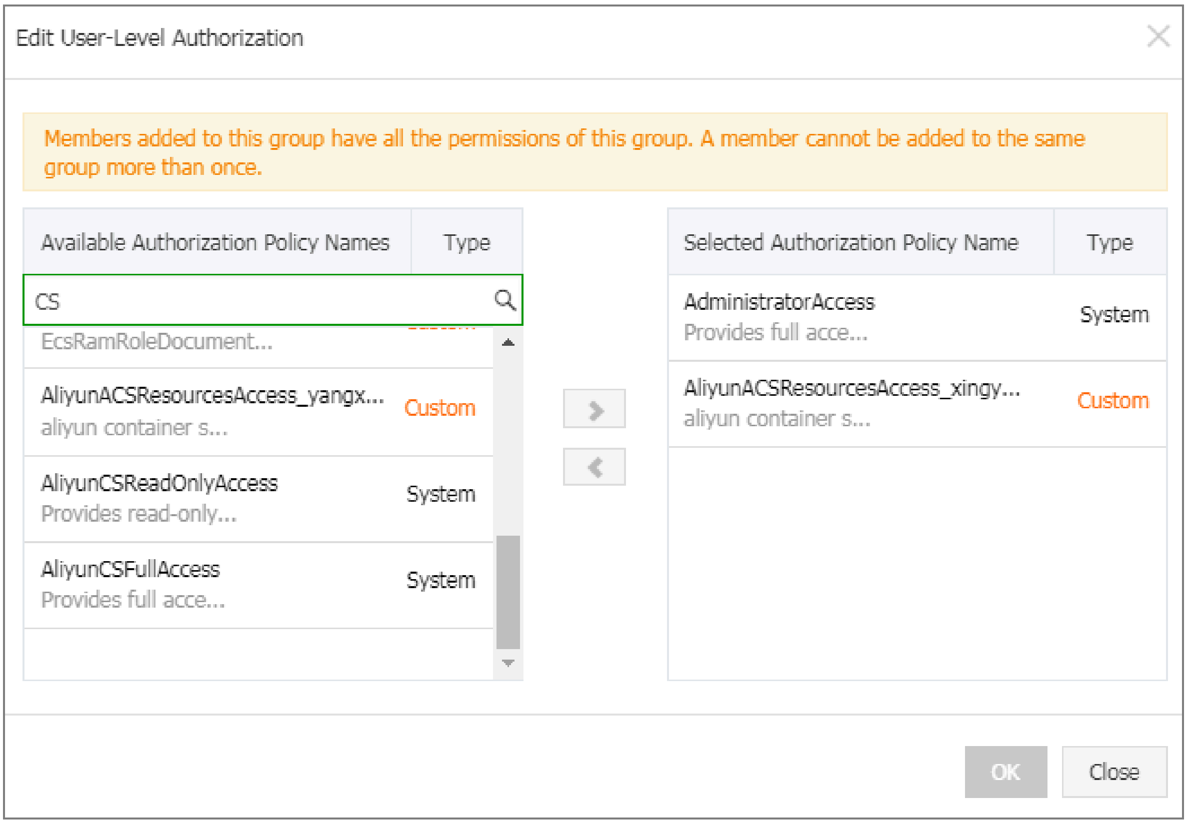
3. Enter a user name for the RAM user and then click OK.
4. On the Users page, select the created RAM user and click Manage.
5. In the Web Console Logon Management area, click Enable Console Logon.
6. Enter a logon password and click OK.

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

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



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



You can use the following system policies:

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

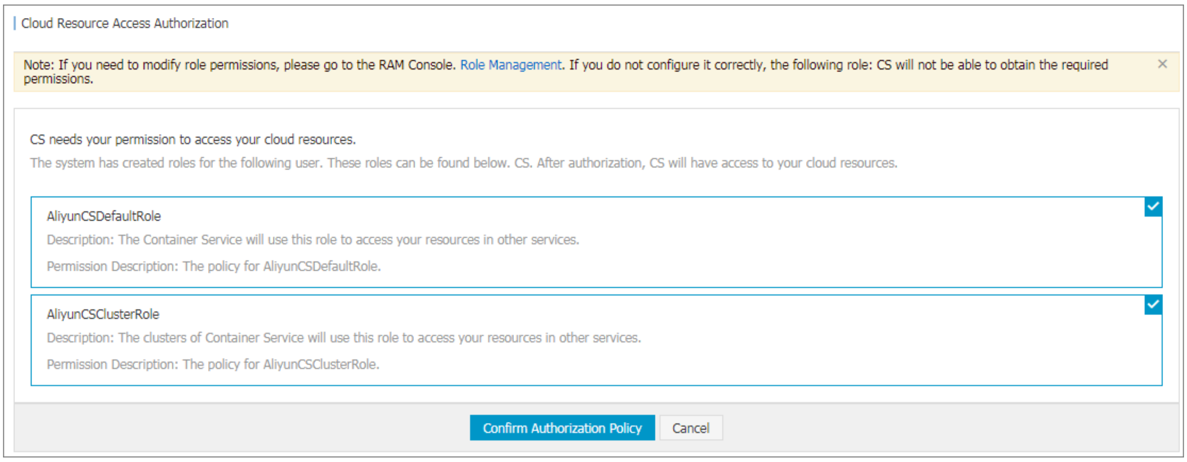
You can also create custom policies as you need and attach them to the RAM user. For more information, see *Create custom authorization policies*.

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

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

  Log on to the *Container Service console* as a RAM user.
- If you have not granted the AliyunCSDefaultRole and AliyunCSClusterRole roles to the Alibaba Cloud account, you must log on to the Container Service console using the account credentials and

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



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

1.3.3 Create custom authorization policies

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

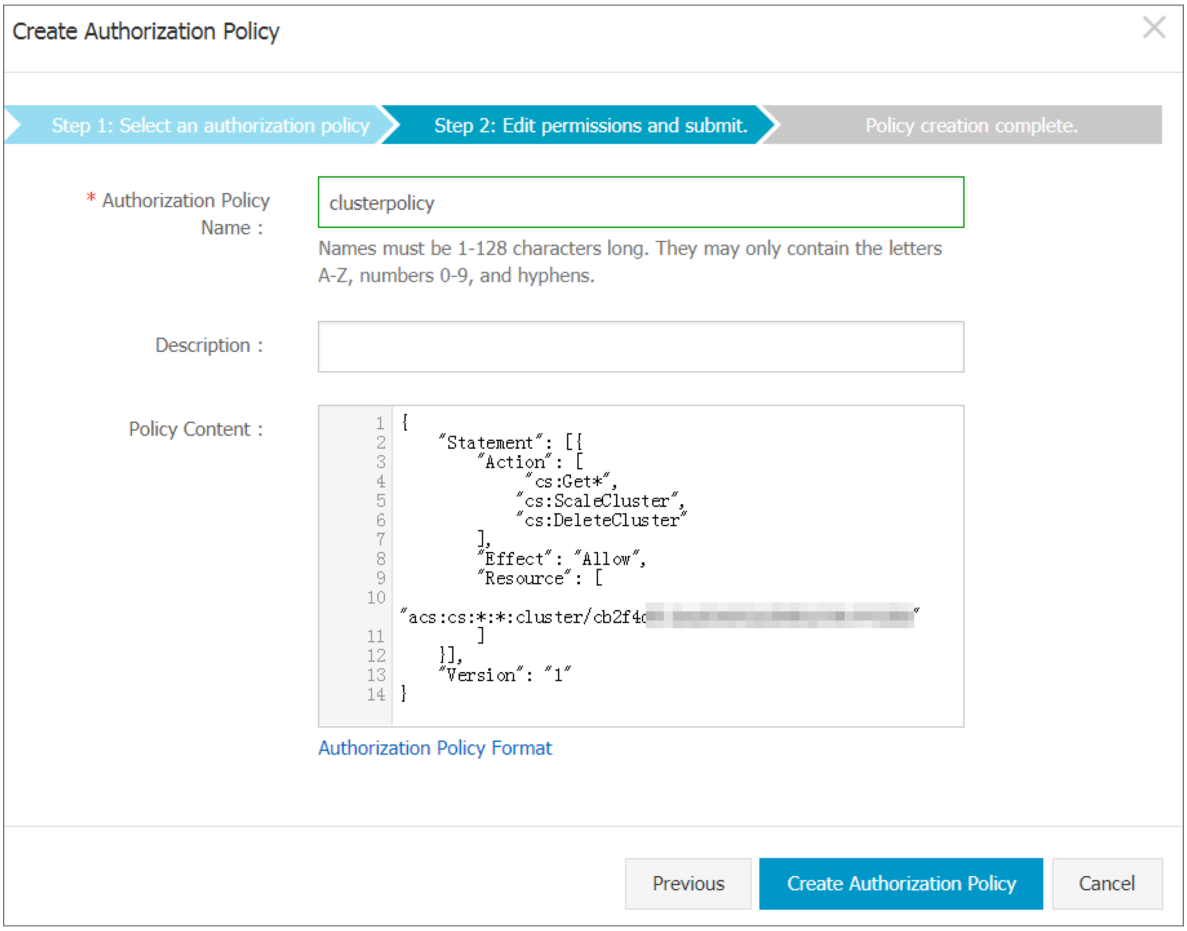
Create custom authorization policies

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

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

Procedure

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




```
{
 "Statement": [{
     "Action": [
         "cs:Get*",
         "cs: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.
GetClusterLogs	View cluster operation logs.
GetClusterEndpoint	View cluster access point.

Action	Description
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.

1.3.4 Kubernetes permission configuration guide for RAM users

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

Prerequisites

- Because the RAM user authorization page is visible only to Alibaba Cloud accounts , you must make sure that you have an Alibaba Cloud account and have created one or several RAM users.
- Due to the security restrictions of Alibaba Cloud RAM, when any RAM authorizat ion is involved in the process of permission configuration through the Container

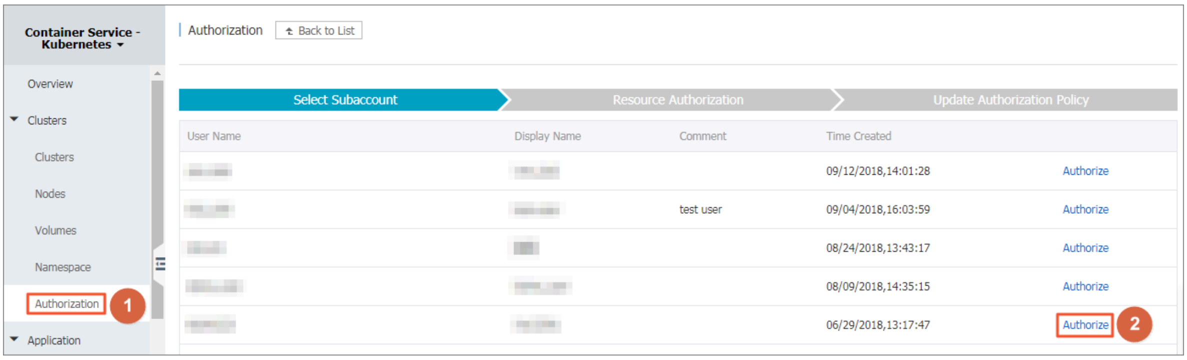
Service console, you must manually perform authorization in the RAM console for the target RAM user according to the reference policy and operation instructions on the page.

Procedures

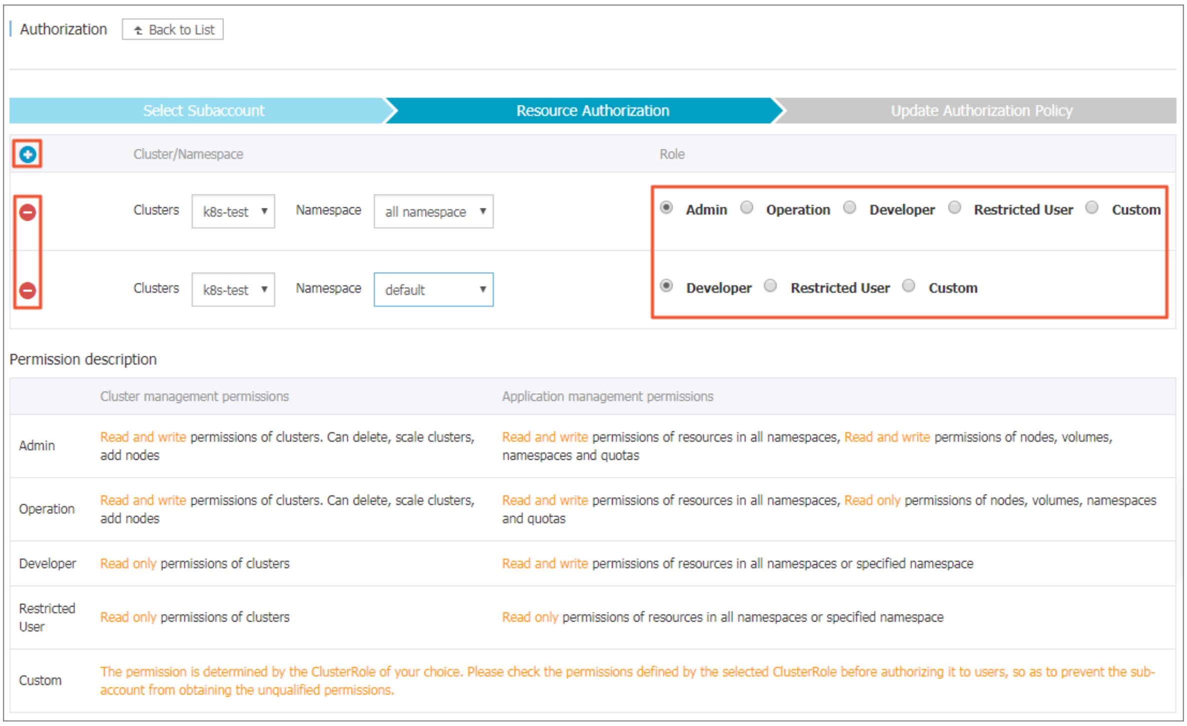
Note:

Because the RAM user authorization page is visible only to Alibaba Cloud accounts, you must use the credentials of your Alibaba Cloud account to log on to the Container Service console.

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, select Container Service - Kubernetes. Then, click Clusters > Authorization to go to the Authorization page.
3. From the sub-account list, find the target RAM user, and click Authorize.



4. On the Resource Authorization page, you can add permissions of the cluster or namespace level by clicking the plus sign in the upper left corner and then selecting a role. You can also click the minus sign to remove the permission.



For information about the definitions of the roles in clusters and namespaces, see the following table.

Table 1-2: Roles and permissions

	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 permission s 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 the resources in all namespaces or specified namespaces.

	Cluster management permissions	Application management permissions
Restricted User	Read-only permissions of clusters.	Read-only permission s of resources in all namespaces or specified namespaces.
Custom	Read and write permissions of clusters . Delete clusters, scale clusters, and add nodes.	The permissions of the RAM user depend on the cluster role you select. Confirm the permissions that your selected cluster role has on resources before authorization, to avoid inappropriate permissions granted to the RAM user.

5. After the preceding configuration is completed, if any change of the RAM permissions of the RAM user is involved, reference configuration for the corresponding Kubernetes cluster RAM permission is displayed on the Update Authorization Policy page. You can complete the RAM user authorization update in the RAM console according to the instructions on the page.

Additional instructions

To avoid affecting your access to existing Kubernetes clusters as a RAM user, the Container Service console is temporarily compatible with the old cluster access permission control. In a certain period of time, the RAM user can access the Kubernetes clusters without RBAC application permission check. If you are a RAM user, contact the Alibaba Cloud account owner in time for authorization according to the following cluster type and compatibility method.

For existing clusters created by the RAM user, 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 the RAM user obtains no authorization from the Alibaba Cloud account or gets no permission management update, the RAM user is banned from accessing the application console corresponding to the cluster.

Table 1-3: Compatible cluster description

Compatible cluster type	Compatibility method
Existing clusters created by a RAM user	A permission management upgrade notice is prompted and a one-click upgrade link is provided. The RAM user can obtain application authorization by clicking the upgrade link.
Existing clusters with access authorized through RAM	A permission management upgrade notice is prompted. Contact the Alibaba Cloud account owner to complete application authorization.
Newly created clusters with access authorized through RAM	A permission management upgrade notice is prompted. Contact the Alibaba Cloud account owner to complete application authorization.

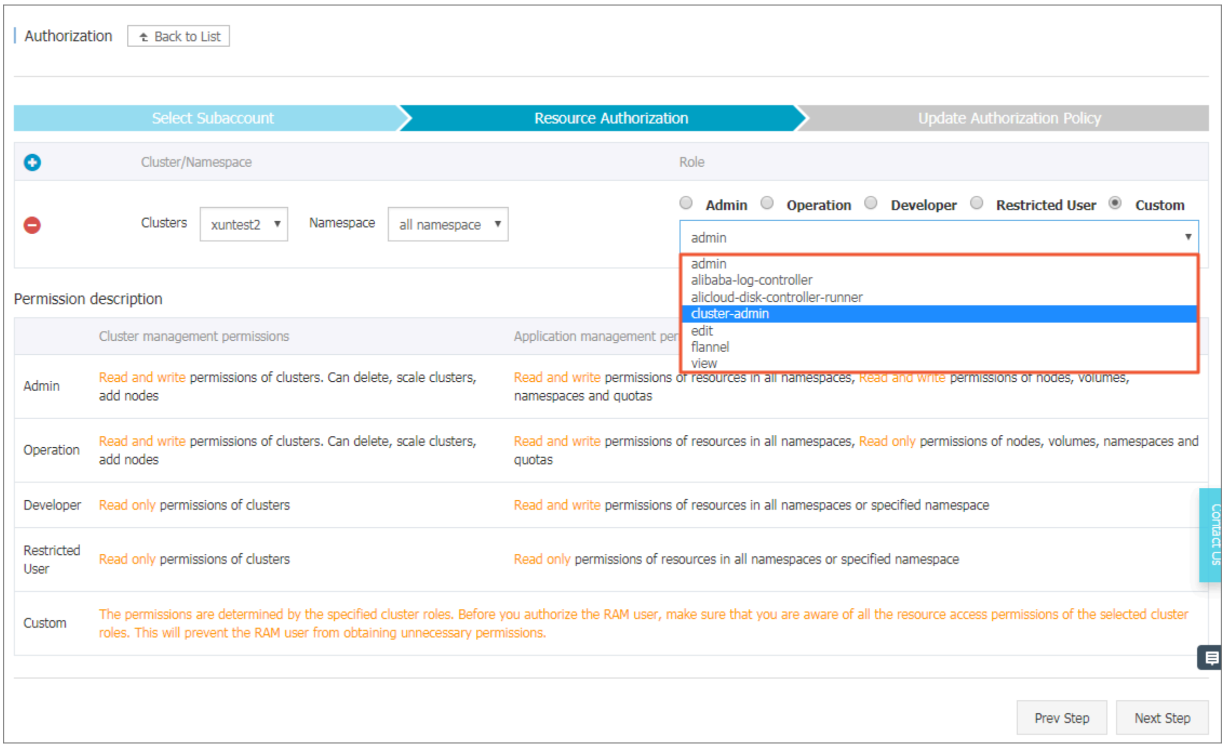
Custom permissions

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

Alibaba Cloud Container Service provides some custom permissions.

Note:

Among them, the cluster-admin permission is a super administrator permission. It has the permissions to access all resources by default.



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

Note:

Only some of the cluster roles are displayed.

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

```
....
system:volume-scheduler
  13d
view
  13d
```

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

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

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

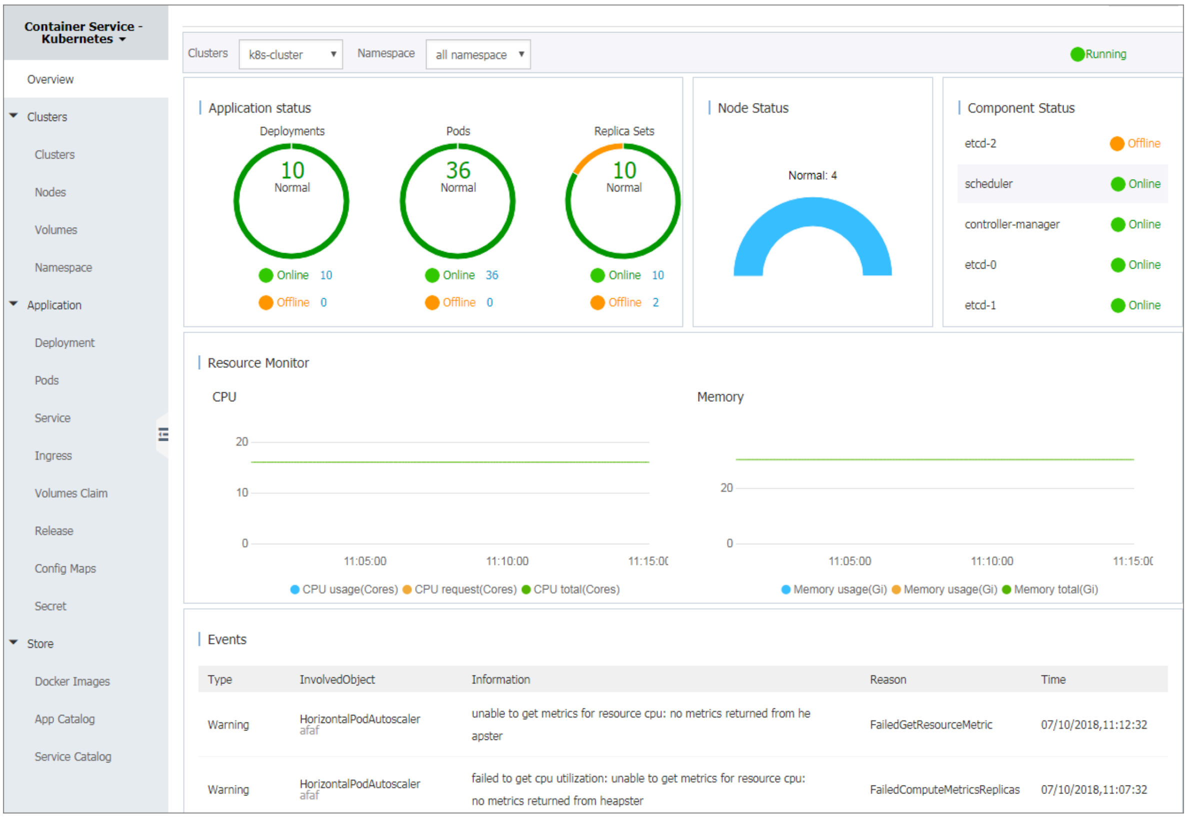
1.4 Cluster management

1.4.1 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.



1.4.2 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.

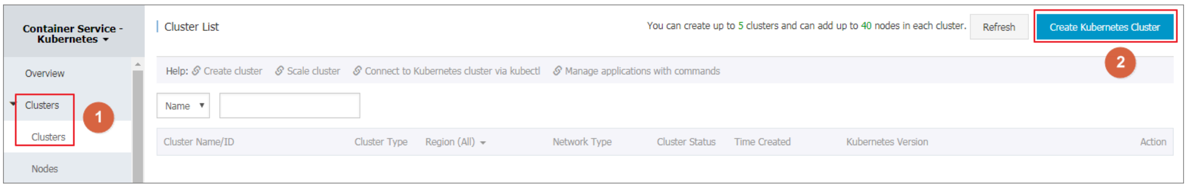
    > Note:
    > In a Kubernetes cluster, the maximum number of default VPC routs is 48, that is, the Kubernetes cluster has up to 48 nodes by default when using VPC. To increase the number of nodes, first open a ticket for the target VPC so as

to increase the number of VPC routes, and then open a ticket for Container Service.

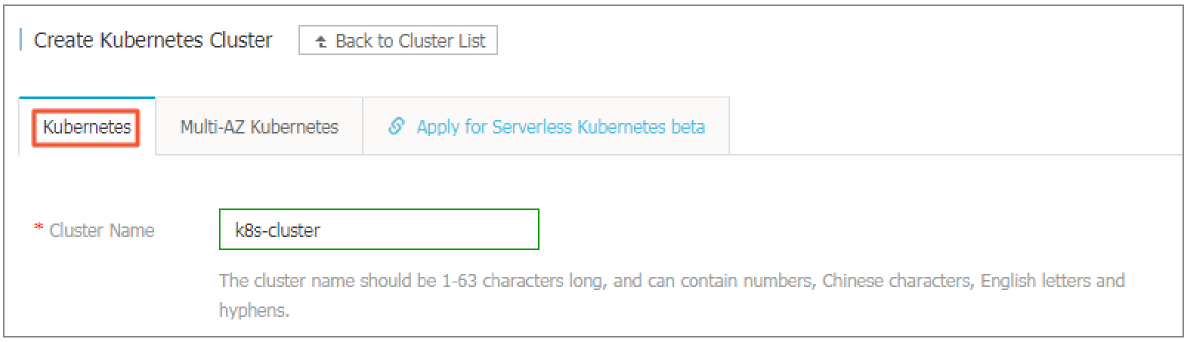
  - By default, each account can create up to 100 security groups.
  - By default, each account can create up to 60 Pay-As-You-Go SLB instances.
  - By default, each account can create up to 20 EIPs.
- The limits for ECS instances are as follows:
  - Only the CentOS operating system is supported.
  - The 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.

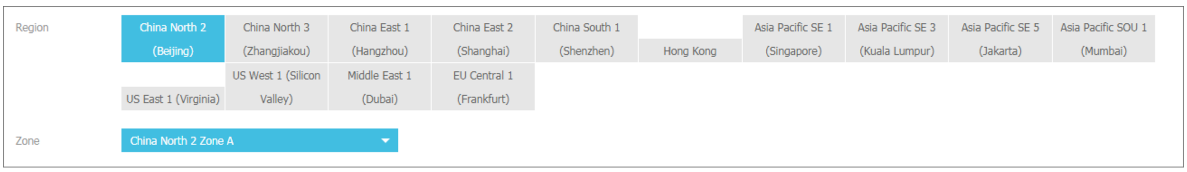


By default, the Create Kubernetes Cluster page is displayed.



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.

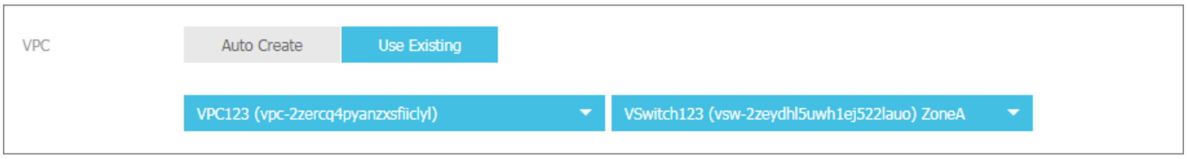


6. Set the cluster network type. Kubernetes clusters support only the VPC network type.

   VPC: You can select Auto Create to create a VPC together with the Kubernetes cluster, or selectUse 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.

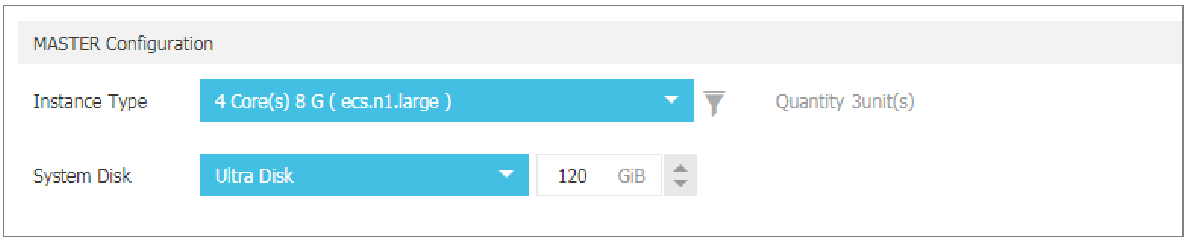


7. Set the node type. Pay-As-You-Go and Subscription types are supported.
8. Configure the Master nodes.

   Select the instance type for the Master nodes.

   Note:

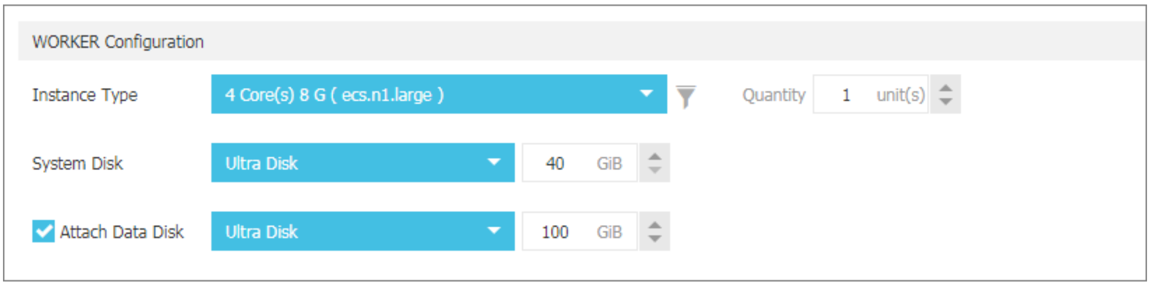
   - Currently, only the CentOS operating system is supported.
   - Currently, you can create only three Master nodes.
   - System disks are attached to the Master nodes by default. Available system disks are SSD Cloud Disks and Ultra Cloud Disks.



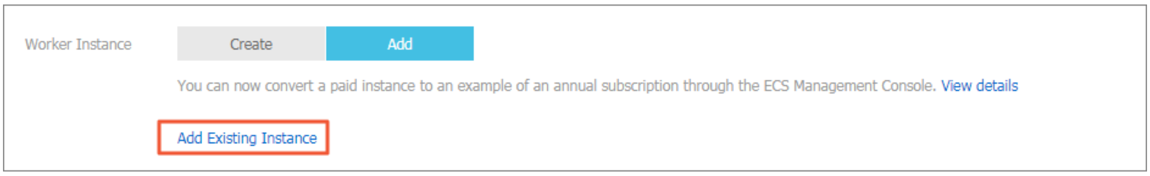
9. Configure the Worker nodes. You can create Worker nodes or add existing instances.

> **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.



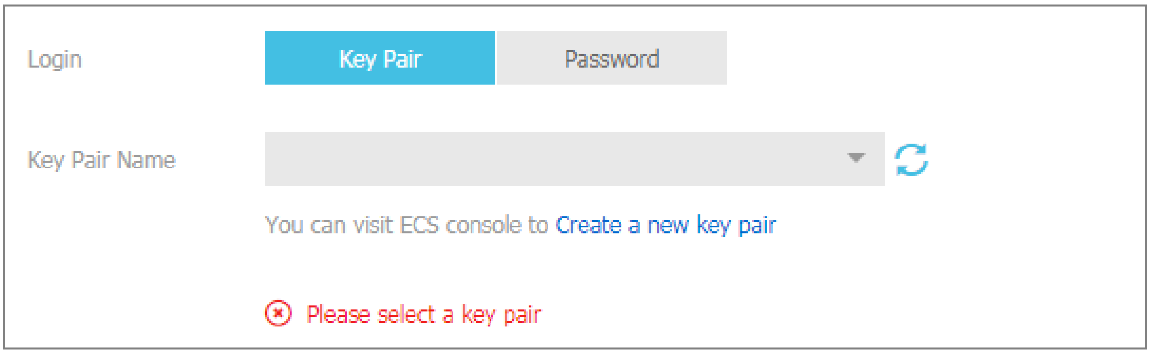
b. To add existing instances, you must create ECS instances in the current region in advance.



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.



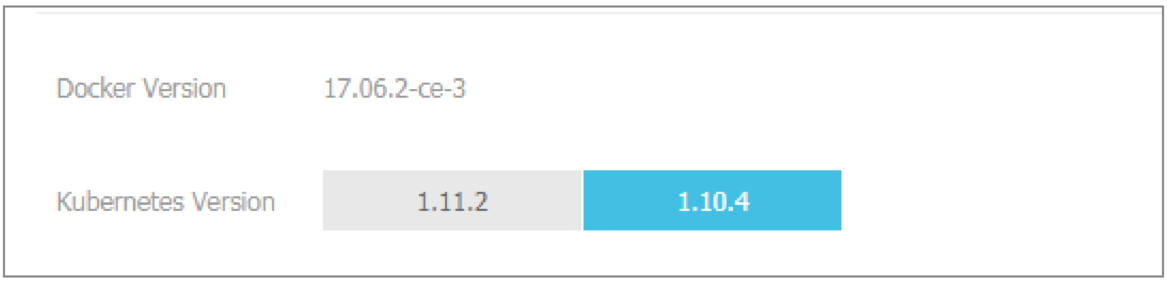
- Set the password.
  - Logon Password: Set the node logon password.
  - Confirm Password: Confirm your node logon password.

11.Set thePod 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.



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

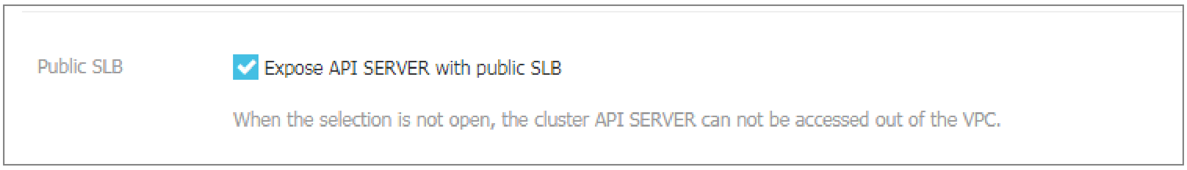
> Note:
> If you select Auto Create, you must configure a SNAT Gateway. If you select Use Existing, you can select whether to automatically configure a SNAT Gateway. If you

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.

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).

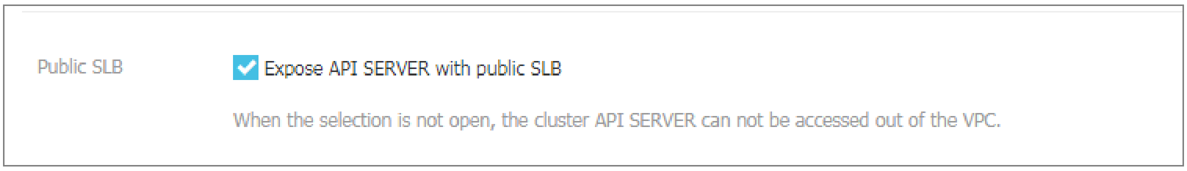
a. If you select to enable this option, the Internet SLB is created and the port 6443 of the Master nodes is exposed. The port corresponds to the API server. Then you can use kubeconfig to connect to and operate the clusters through the Internet.
b. If you select not to enable this option, the Internet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



15. Select whether to enable SSH logon for Internet.

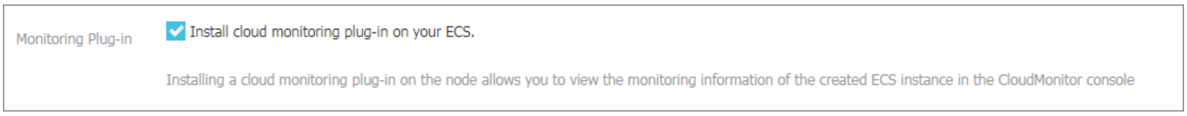
Note:
To enable SSH access for Internet, you must select Use Public SLB to Expose API Server.

- 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*.



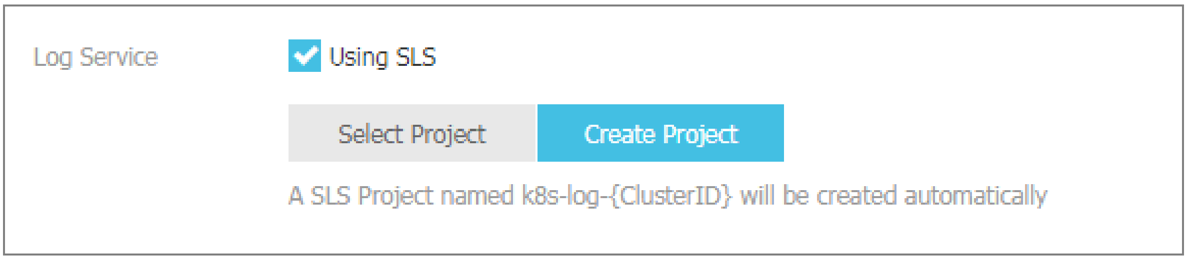
16.Select whether to install a cloud monitoring plug-in on your ECS.

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.



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*.



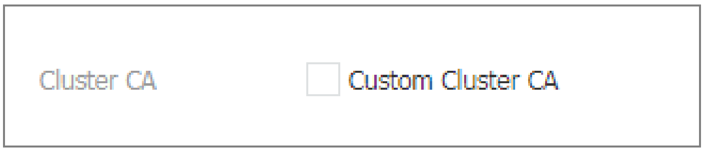
18.Select whether to show advance config.

a. Select a network plug-in. Available network plug-ins are Flannel and Terway. For details, see *Do I select the Terway or Flannel plugin 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.

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.



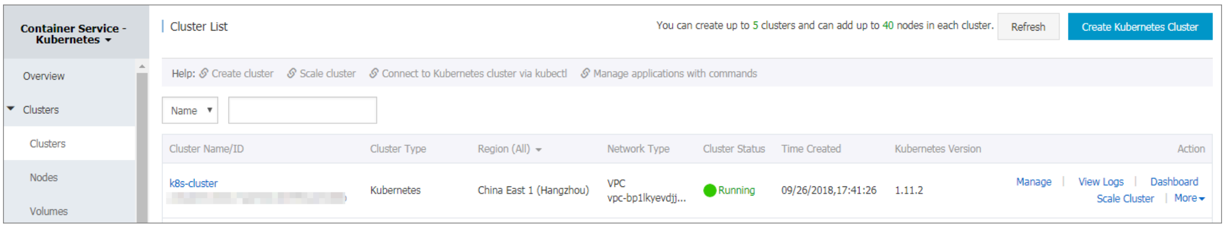
19.Click Create in the upper-right corner.

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.

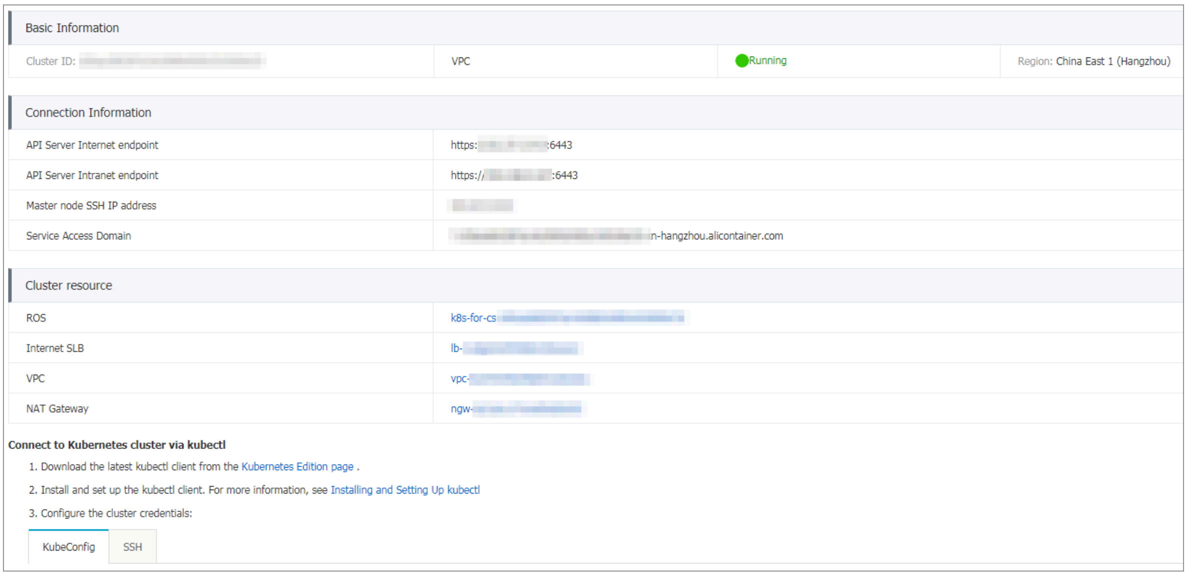


- Click View Logs on the right of the cluster to view the cluster logs. To view more detailed information, click Stack Events.

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.

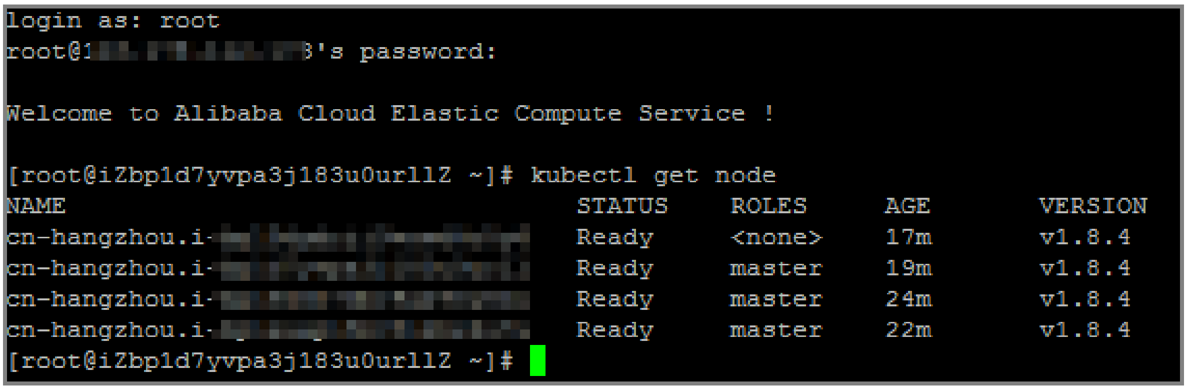


In the Cluster Info section:

- 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@[redacted]'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.

1.4.3 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

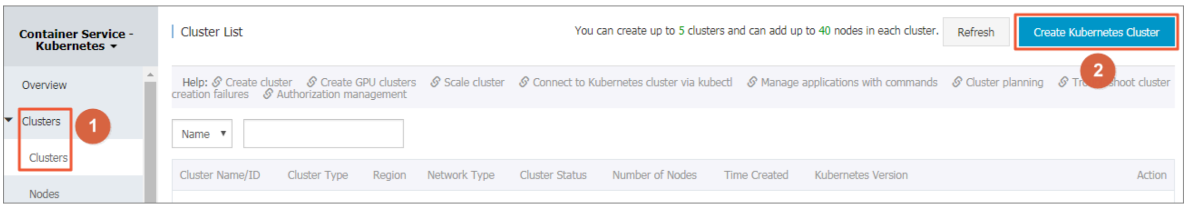
- The SLB instance created with the cluster only supports the Pay-As-You-Go billing method.

- 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.

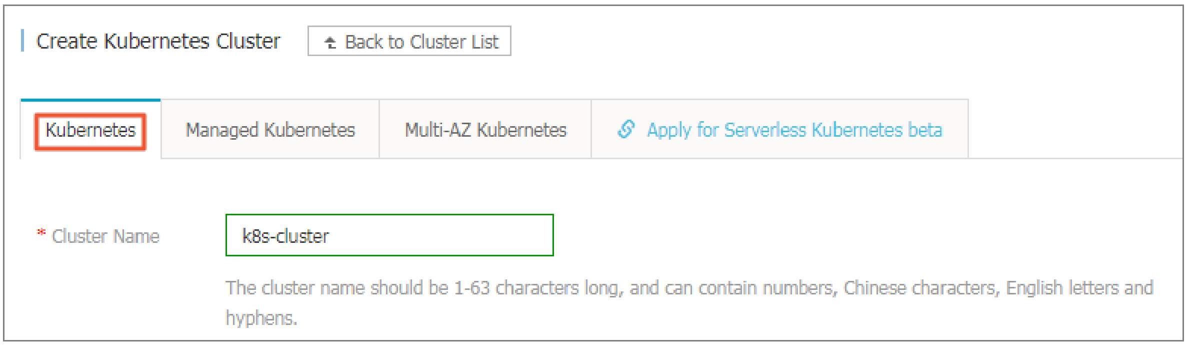
Create a GN5 Kubernetes cluster

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.

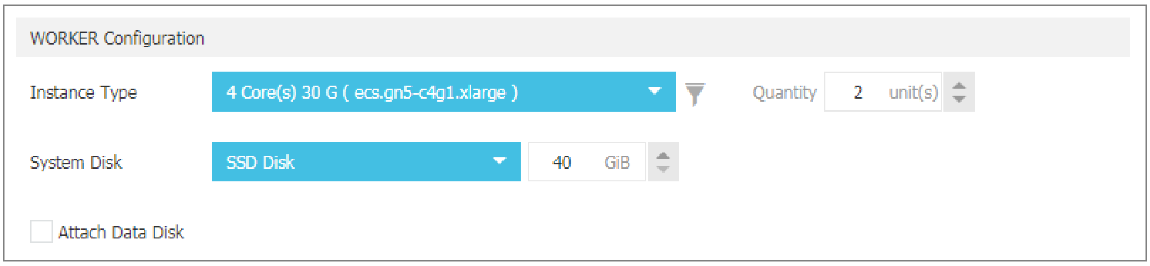


By default, the Create Kubernetes Cluster page is displayed.

> 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*.



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

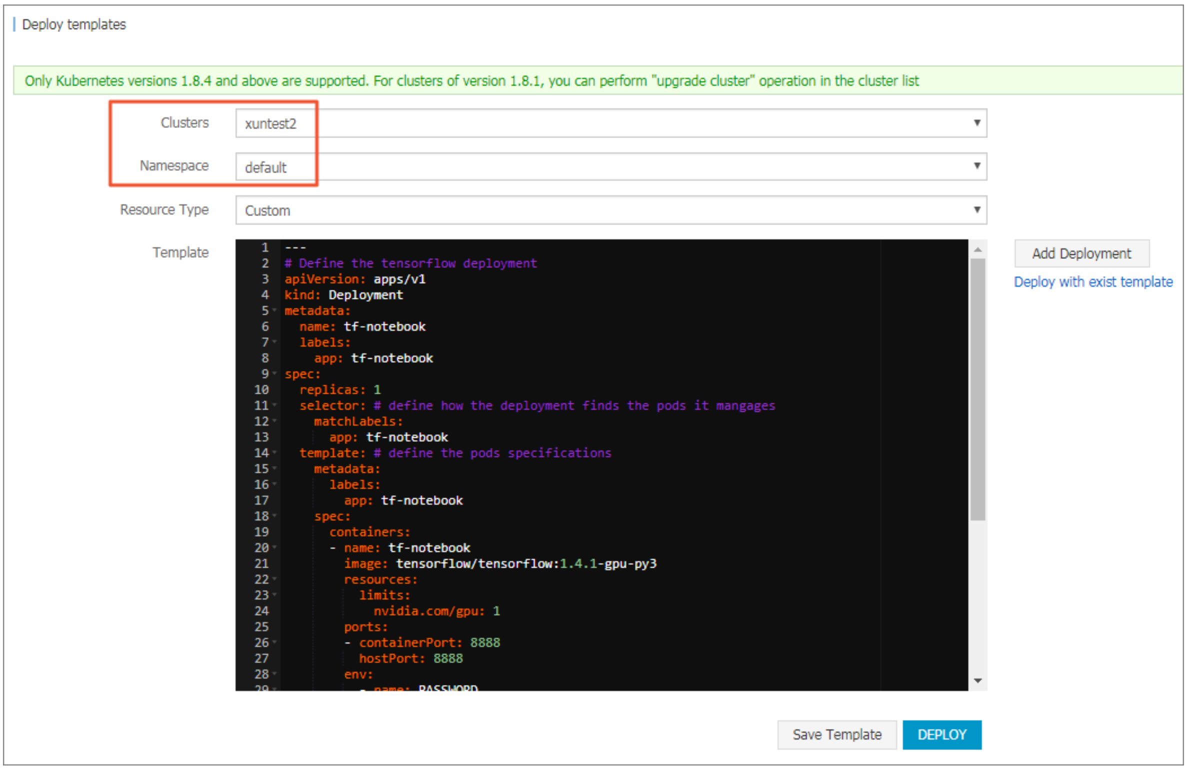
    b. If you choose to add existing instances, you need to have already created GPU cloud servers in the same region where the cluster is to be created.
5. After you have completed all required settings, click Create to start cluster deployment.
6. After the cluster is created, choose Clusters > Nodes in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

Create a GPU experimental environment to run TensorFLow

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

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



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
```

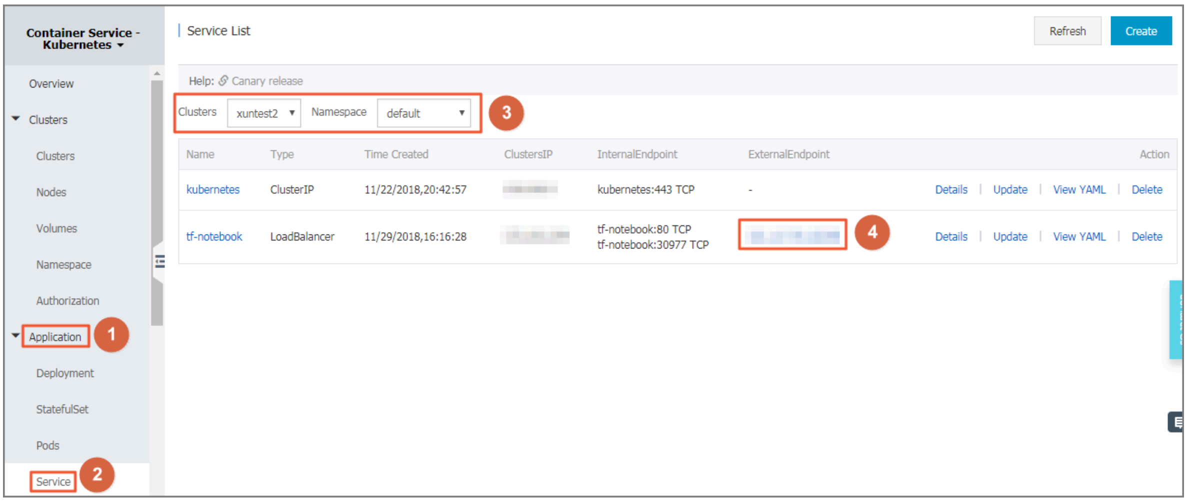
```
          hostPort: 8888
        env:
          - name: PASSWORD                         #specify the
password used to access the Jupyter service. You can modify the
password as needed.
            value: mypassw0rd

# 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
```

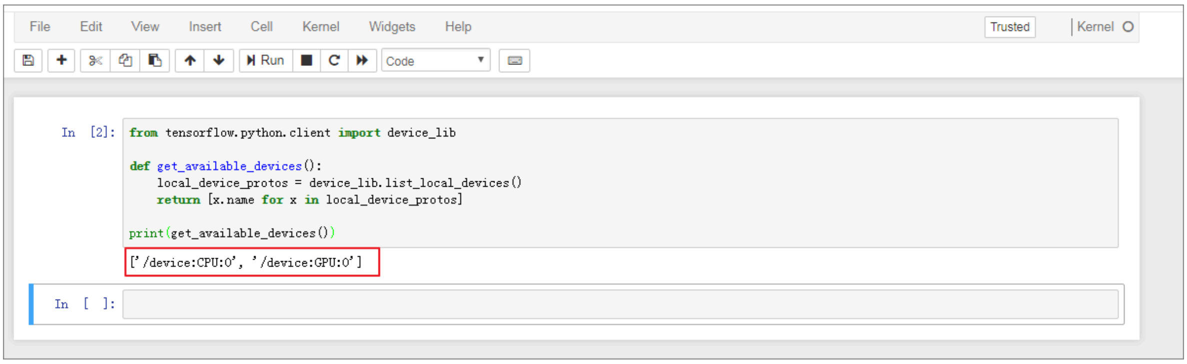
When you orchestrate your deployment template in a cluster by using the GPU deployment solution of Kubernetes earlier than 1.9.3, your template must be highly dependent on the cluster. As a result, portability of the template is not achievable. However, in Kubernetes version 1.9.3 and later, you do not 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.



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())
```



1.4.4 Upgrade the NVIDIA driver of a Kubernetes cluster GPU node

This topic describes how to upgrade the NVIDIA driver of a Kubernetes cluster GPU node where services are running or no service runs.

Prerequisites

- You have created a Kubernetes GPU cluster. For more information, see *Configure a Kubernetes GPU cluster to support GPU scheduling*.

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

Upgrade the NVIDIA driver of a GPU node where services are running

1. Run the following command to disable scheduling for the target GPU node:

```
kubectl cordon node-name
```

> **Note:**
>
> - Only the NVIDIA drivers of Worker nodes can be upgraded.
> - The `node-name` parameter must be in the format of `your-region-name.node-id`.
>   - `your-region-name` indicates the name of the region where your cluster is located.
>   - `node-id` indicates the ID of the ECS instance where the target node is located.
>
>   You can run the following command to view `node-name`:
>
> ```
> kubectl get node
> ```

```
[root@gpu-test ~]# kubectl cordon cn-hangzhou.i-[redacted]
node/cn-hangzhou.i-[redacted] already cordoned
```

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

```
kubectl drain node-name --grace-period=120 --ignore-daemonsets=true
```

```
[root@gpu-test ~]# kubectl drain cn-hangzhou.i-[redacted] --grace-period=120 --ignore-daemonsets=true
node/cn-hangzhou.i-[redacted] cordoned
WARNING: Ignoring DaemonSet-managed pods: flexvolume-[redacted], kube-flannel-ds-[redacted], kube-proxy-worker-[redacted], logtail-ds-[redacted]
pod/domain-nginx-[redacted] evicted
pod/old-nginx-[redacted] evicted
pod/new-nginx-[redacted] evicted
pod/old-nginx-[redacted] evicted
```

3. Run the following command to log on to the target node:

```
ssh root@xxx.xxx.x.xx
```

4. On the target node, run the following command to view the driver version:

```
nvidia-smi
```

```
[root@[redacted] ~]# nvidia-smi
Fri Jan 18 16:44:52 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 384.111                Driver Version: 384.111                   |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P4            On   | 00000000:00:08.0 Off |                    0 |
| N/A   24C    P8     6W /  75W |      0MiB /  7606MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

5. Run the following commands to remove the existing driver:

> **Note:**
>
> - If the existing driver is v`384.111`, you can directly run the commands in this step.
> - If the existing driver is not v`384.111`, you must download the correct driver version from NVIDIA Website before running the commands in this step.

```
cd /tmp
```

```
curl -O https://cn.download.nvidia.cn/tesla/384.111/NVIDIA-Linux-
x86_64-384.111.run
```

```
chmod u+x NVIDIA-Linux-x86_64-384.111.run
```

```
. /NVIDIA-Linux-x86_64-384.111.run --uninstall -a -s -q
```

6. Run the following command to restart the target node:

```
reboot
```

7. Download the driver version that you want from the NVIDIA Website. This example uses v `410.79`.

8. Run the following command to install the downloaded NVIDIA driver in the directory where the driver is downloaded:

```
sh . /NVIDIA-Linux-x86_64-410.79.run -a -s -q
```

9. Run the following commands to add the following settings to the NVIDIA driver:

```
nvidia-smi -pm 1 || true
```

```
nvidia-smi -acp 0 || true
```

10. Run the following command to update two device plugins:

```
mv /etc/kubernetes/manifests/nvidia-device-plugin.yml /
```

```
mv /nvidia-device-plugin.yml /etc/kubernetes/manifests/
```

11. In any path of the Master node, run the following command to enable scheduling for the target node:

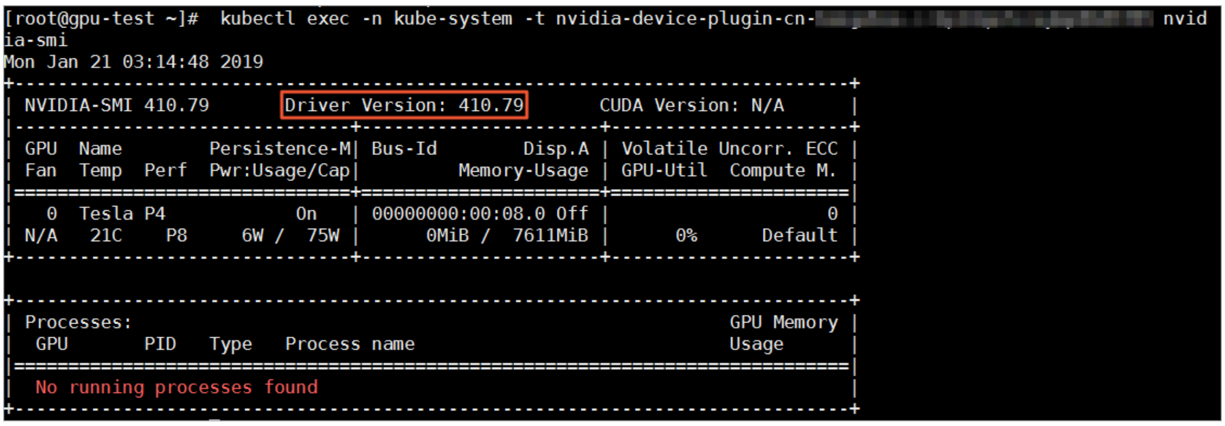
```
kubectl uncordon node-name
```

Verify the results

Run the following command on the Master node. Then check the driver version for the target GPU node. The system displays that the driver is `v410.79`, indicating the node driver has been upgraded.

Note:
You need to replace the *node-name* parameter with your target node name.

```
kubectl exec -n kube-system -t nvidia-device-plugin-node-name nvidia-smi
```



Upgrade the NVIDIA driver of a GPU node where no service runs

1. Run the following command to log on to the target GPU node:

```
ssh root@xxx.xxx.x.xx
```

2. On the target node, run the following command to view the driver version:

```
nvidia-smi
```

```
[root@[redacted] ~]# nvidia-smi
Fri Jan 18 16:44:52 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 384.111                Driver Version: 384.111                   |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P4            On   | 00000000:00:08.0 Off |                    0 |
| N/A   24C    P8     6W /  75W |      0MiB /  7606MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

3. Run the following commands to remove the existing driver:

> Note:
>
> - If the existing driver is `v384.111`, you can directly run the commands in this step.

- If the existing driver is not `v384.111`, you must download the correct driver version from NVIDIA Website before running the commands in this step.

```
cd /tmp
```

```
curl -O https://cn.download.nvidia.cn/tesla/384.111/NVIDIA-Linux-
x86_64-384.111.run
```

```
chmod u+x NVIDIA-Linux-x86_64-384.111.run
```

```
. /NVIDIA-Linux-x86_64-384.111.run --uninstall -a -s -q
```

4. Run the following command to restart the target node:

```
reboot
```

5. Download the driver version that you want from the NVIDIA Website. This example uses v `410.79`.
6. Run the following command to install the downloaded NVIDIA driver in the directory where the driver is downloaded:

```
sh . /NVIDIA-Linux-x86_64-410.79.run -a -s -q
```

7. Run the following commands to add the following settings to the NVIDIA driver:

```
nvidia-smi -pm 1 || true
```

```
nvidia-smi -acp 0 || true
```

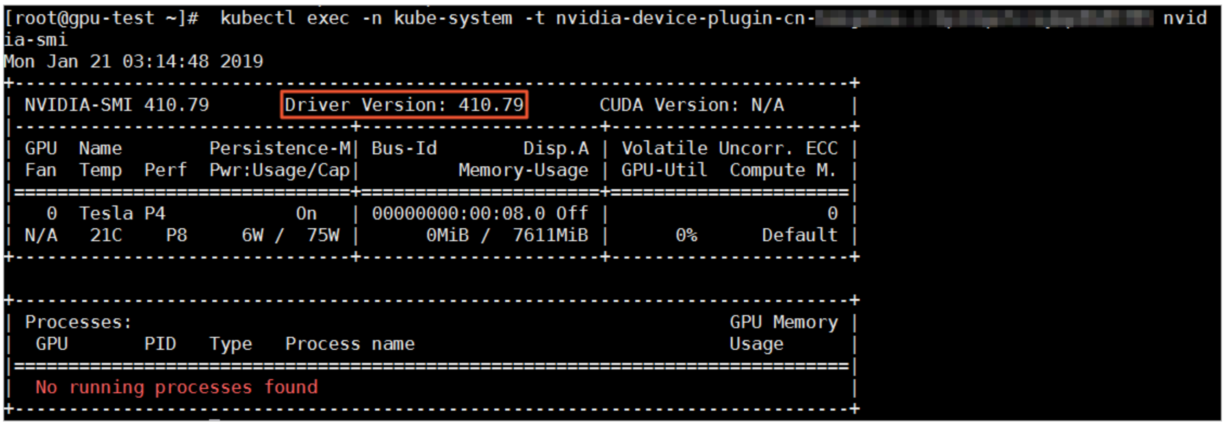
Verify the results

Run the following command on the Master node. Then check the driver version for the target GPU node. The system displays that the driver is `v410.79`, indicating the node driver has been upgraded.

Note:

You need to replace the *node-name* parameter with your target node name.

```
kubectl exec -n kube-system -t nvidia-device-plugin-node-name nvidia-
smi
```



```
[root@gpu-test ~]#  kubectl exec -n kube-system -t nvidia-device-plugin-cn-[redacted] nvid
ia-smi
Mon Jan 21 03:14:48 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.79       Driver Version: 410.79       CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P4            On   | 00000000:00:08.0 Off |                    0 |
| N/A   21C    P8     6W /  75W |      0MiB /  7611MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

1.4.5 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

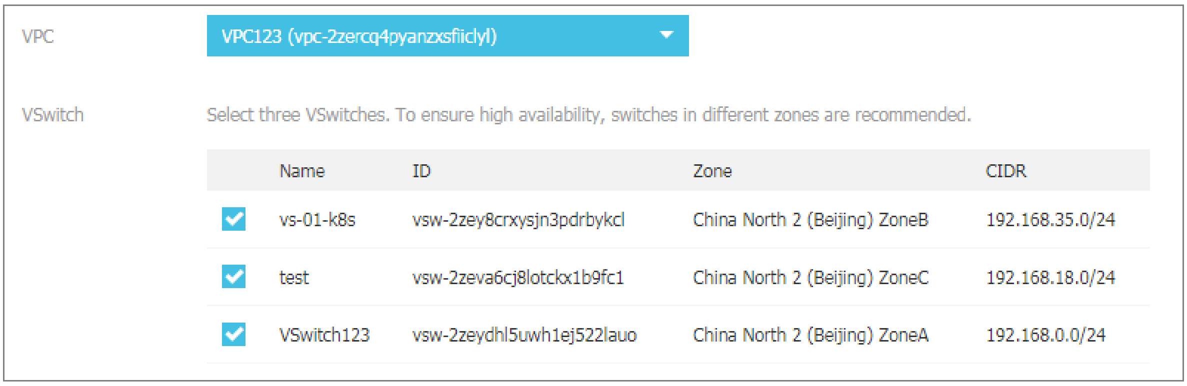
- The SLB instances created with the cluster support only the Pay-As-You-Go billing method.
- Kubernetes clusters support only the Virtual Private Cloud (VPC) network type.
- By default, each account has a specified quota for the cloud resources it can create. If the number of cloud resources exceeds 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.



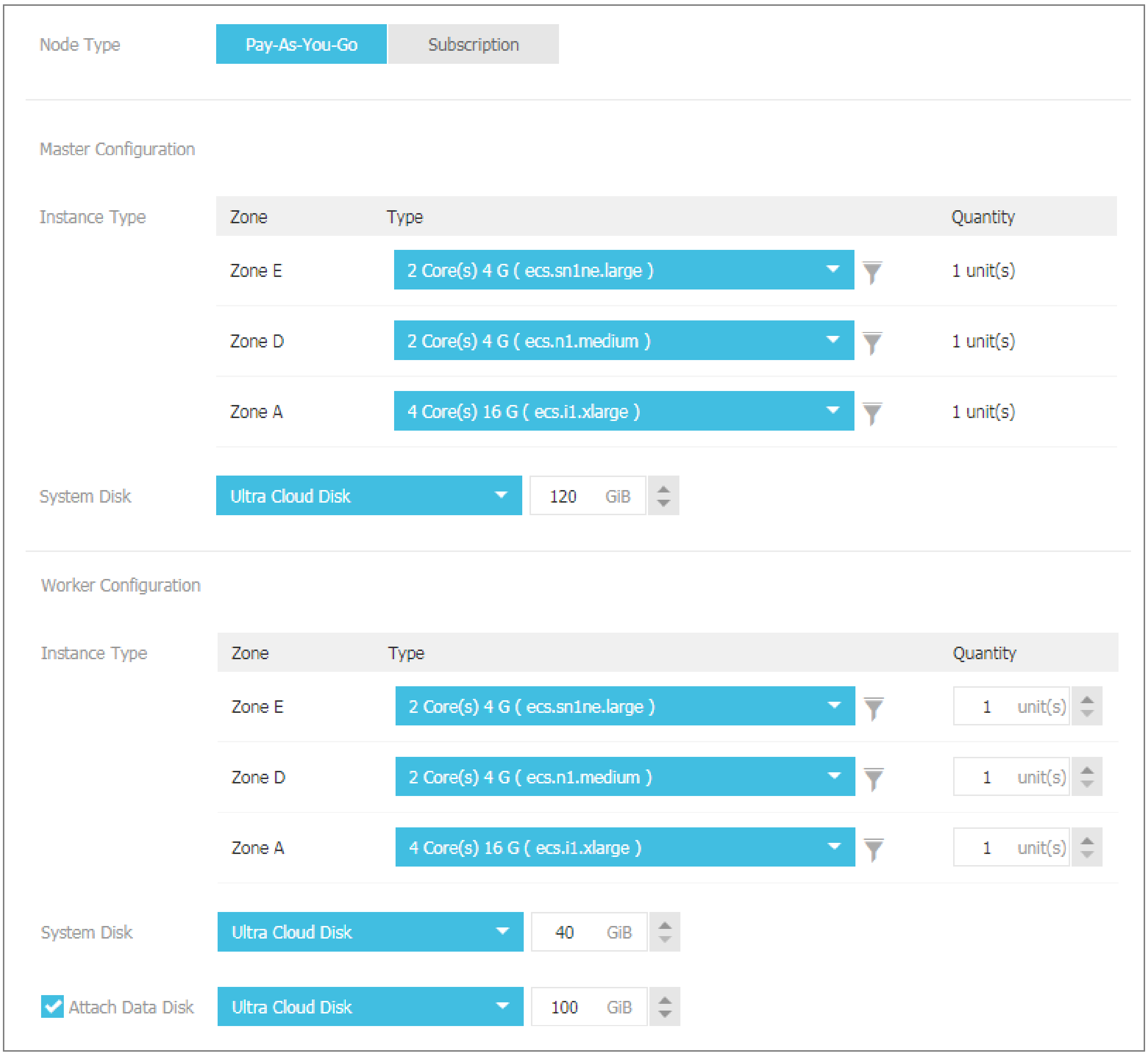
7. Configure the Master nodes and Worker nodes.

   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.

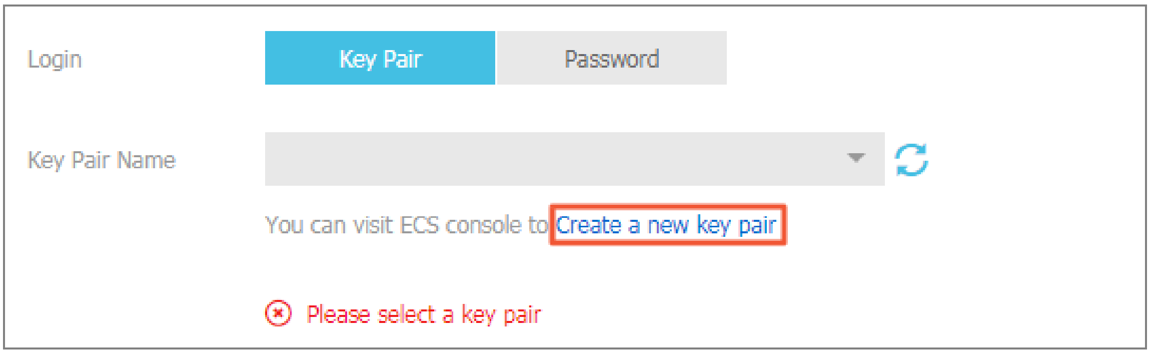
- You can also manually attach a data disk to the Worker node. The data disk can be an Ultra Cloud Disk or an SSD Cloud Disk.



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.

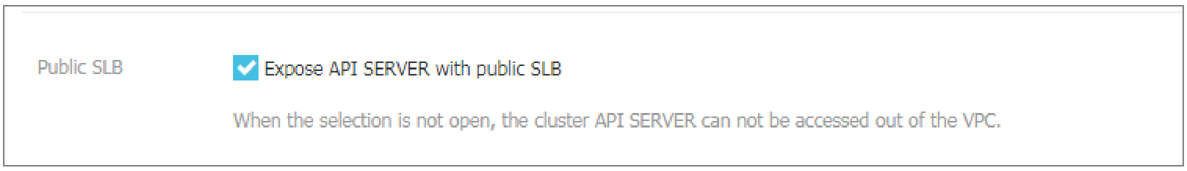
9. Specify the Pod Network CIDR and Service CIDR parameters.

   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).

    a. 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.
    b. 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.



11. Select whether to enable SSH logon for Internet.

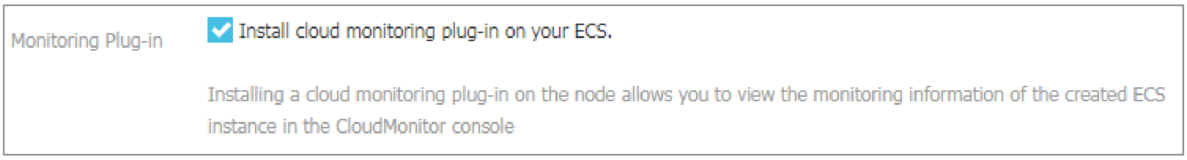
    Note:

    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*.

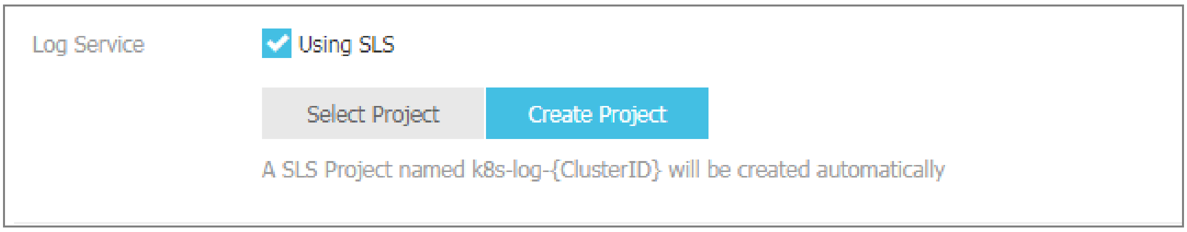
12.Select whether to install a cloud monitoring plug-in on your ECS.

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.



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*.



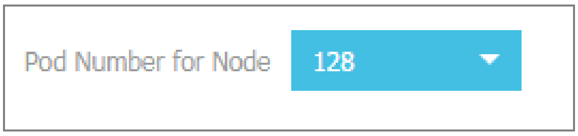
14.Select whether to show advance config.

a. Select a network plug-in. Available network plug-ins are Flannel and Terway. For details, see *Do I select the Terway or Flannel plugin 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.

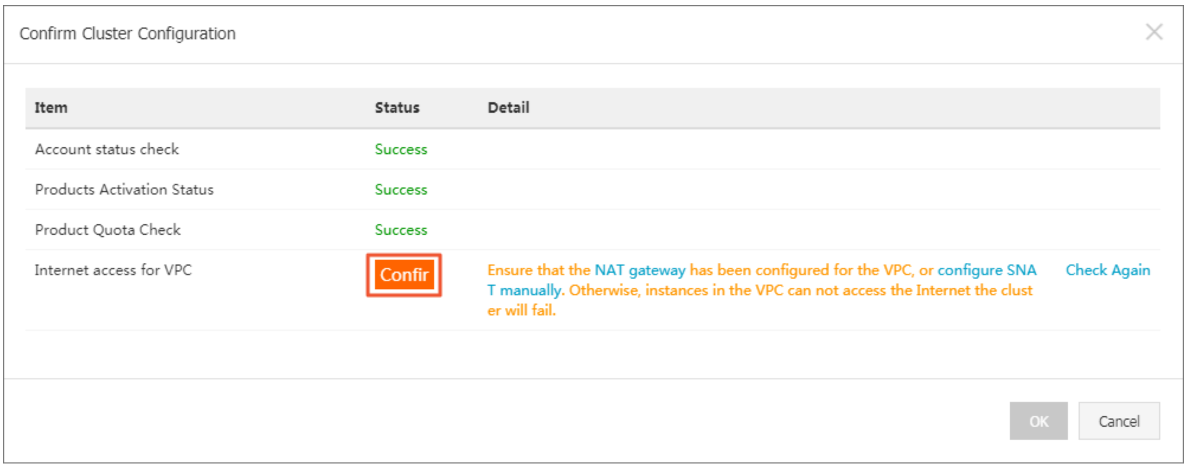


c. Select whether to use Custom Image. The ECS instance installs the default CentOS version if no custom image is selected.

Currently, you can only select an image based on CentOS custom version to quickly deploy the environment you need.

d. 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.

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

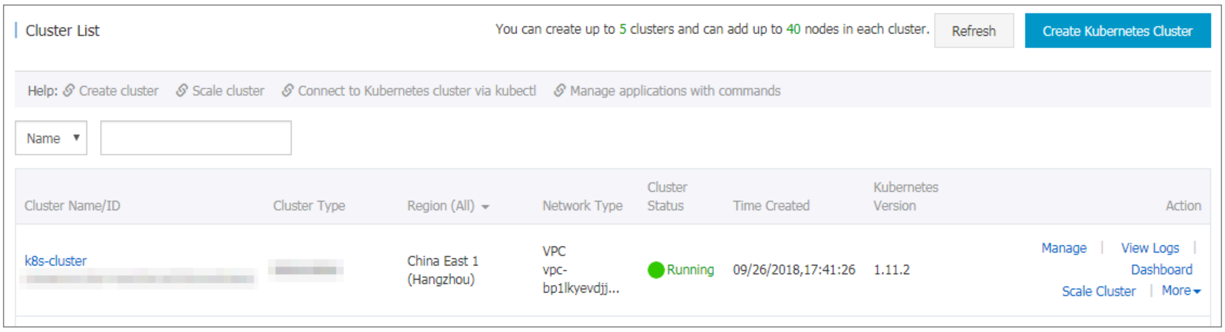


Note:
A multi-node Kubernetes cluster typically takes 10 minutes to be created.

Result

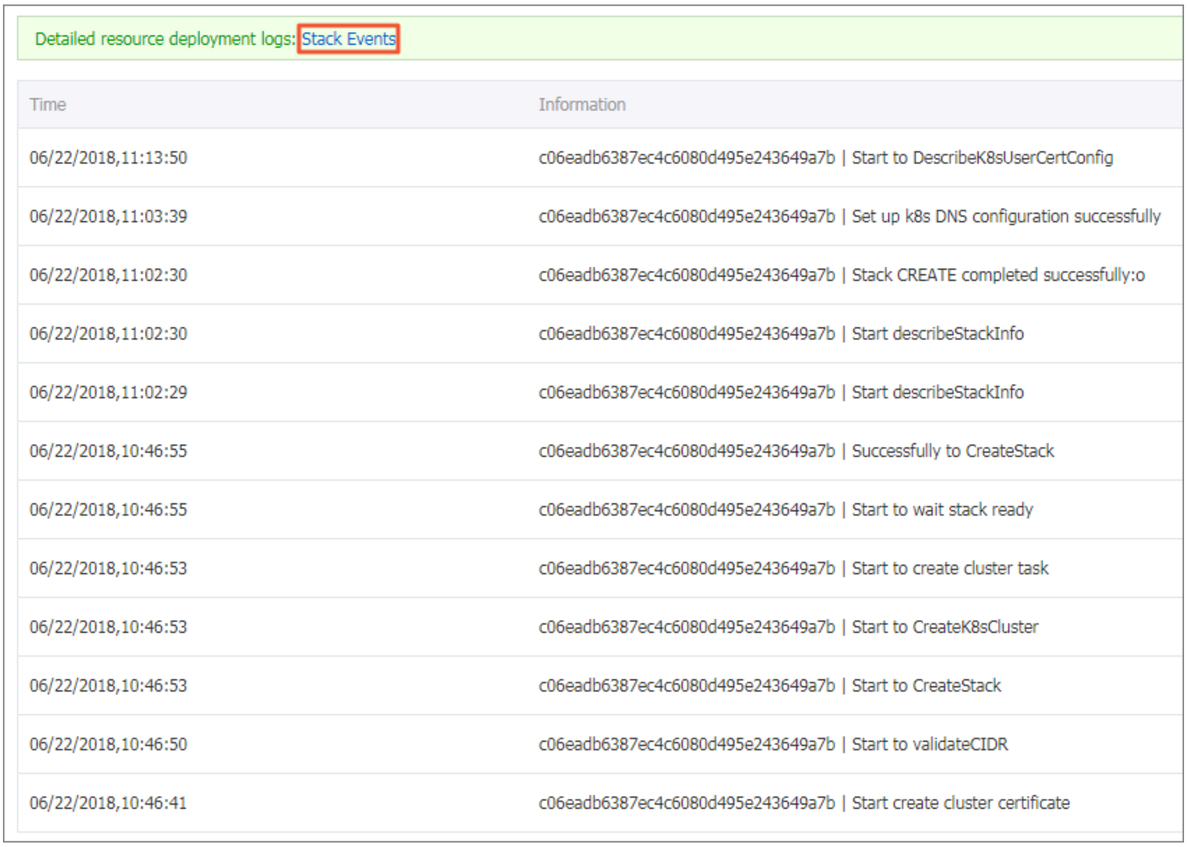
View cluster deployment results.

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



What's next

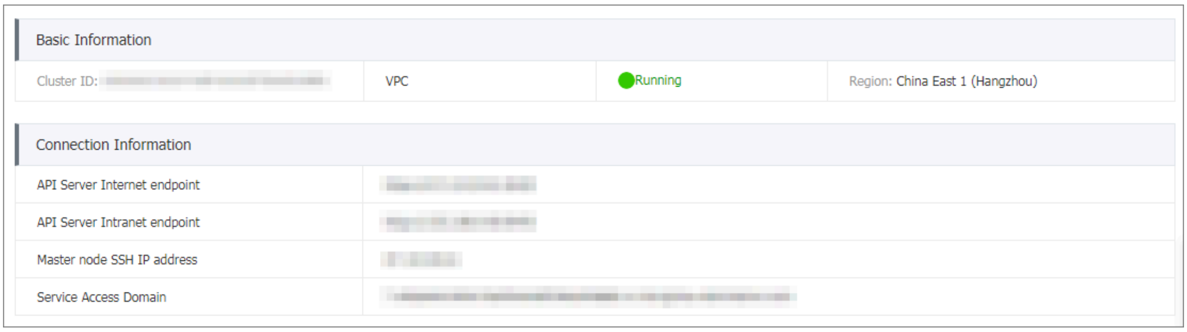
- Click View Logs at the right of the cluster to view the cluster logs. To view more detailed information, click Stack Events.



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.



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`.

1.4.6 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.

1.4.7 Use kubectl on Cloud Shell to manage a Kubernetes cluster

This topic describes how to use kubectl on Cloud Shell to manage a Kubernetes cluster after you log on to the console of Alibaba Cloud Container Service for Kubernetes.

Prerequisites

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

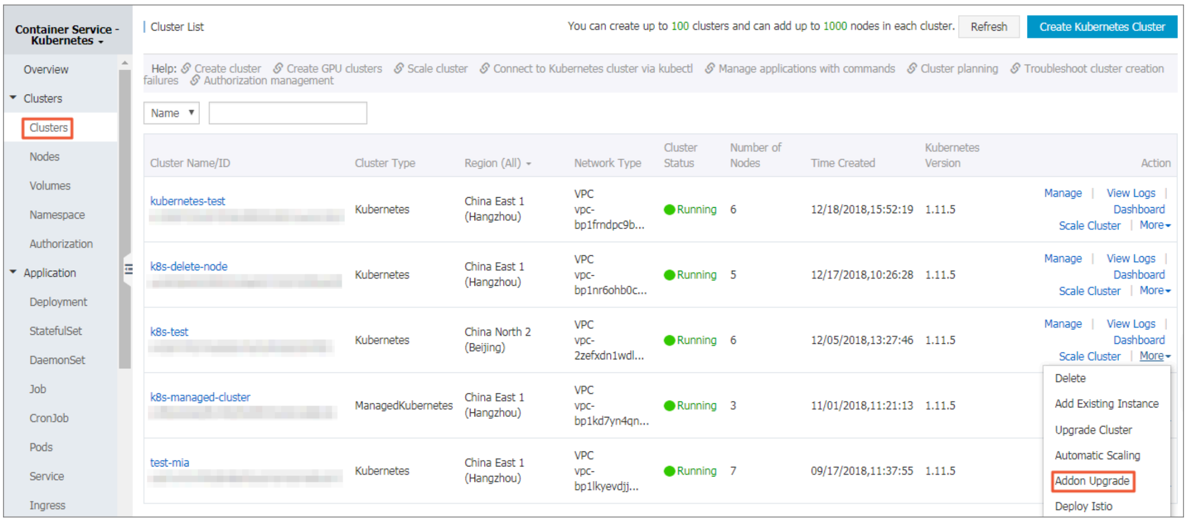
Context

If you want to use kubectl to manage a Kubernetes cluster of Container Servcie, you can download kubectl to your local host. For more information, see *Connect to a Kubernetes cluster by using kubectl*. Additionally, you can also start Cloud Shell on the console of Container Service for Kubernetes, and then use kubectl on Cloud Shell to manage a Kubernetes cluster.

Procedure

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

2. In the left-side navigation pane under Kubernetes, click Clusters.
3. In the Action column of the target cluster, choose More > cos.cls.cloudshell.



Note:

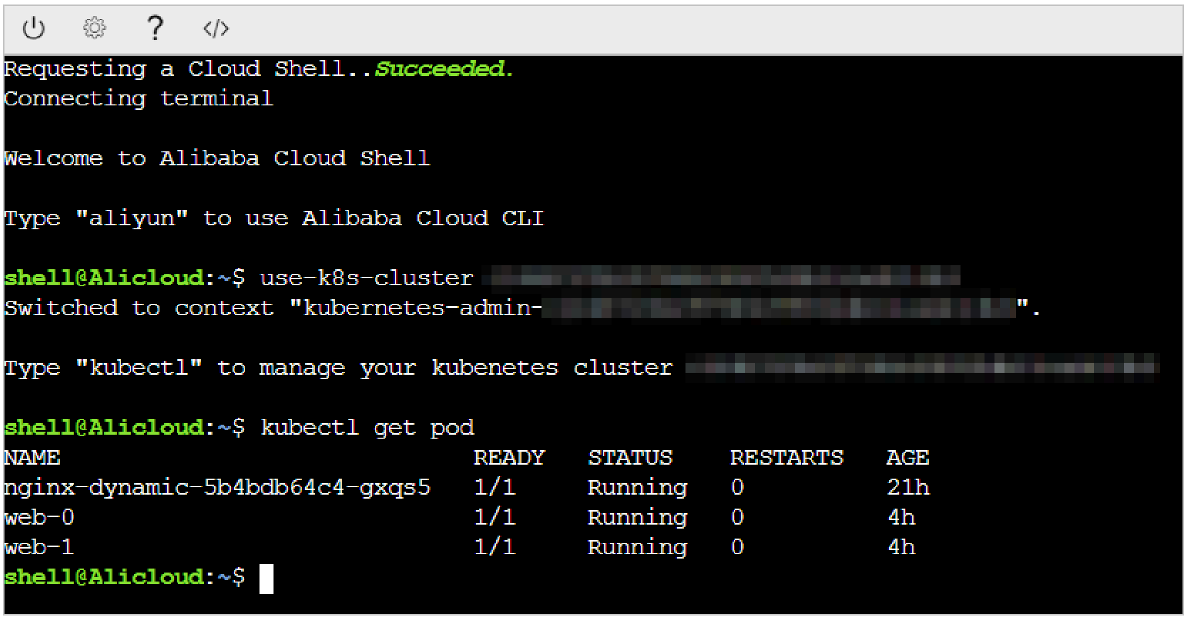
Do the following:

- On the Authorization page, click OK to obtain a temporary access key that expires within one hour.
- On the Storage Space page, click Create Now or Skip as needed.

4. On Cloud Shell, you can use kubectl to manage a Kubernetes cluster of Container Service.

Note:

When you start Cloud Shell associated with the Kubernetes cluster, the system loads the `kubeconfig` file of the cluster onto Cloud Shell. Then you can use kubectl to manage your cluster.



1.4.8 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.

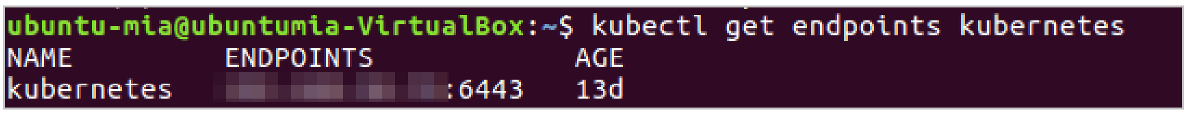
Context

- 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
```



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`:

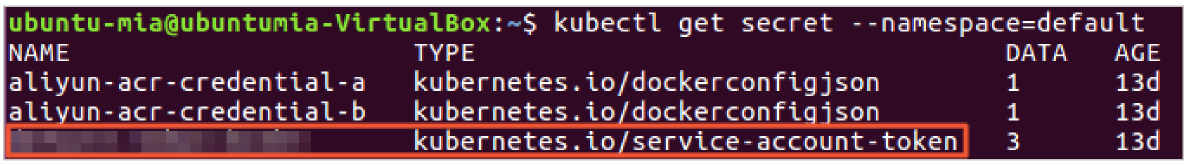
```
$ kubectl get service name
```

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`.

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
```



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

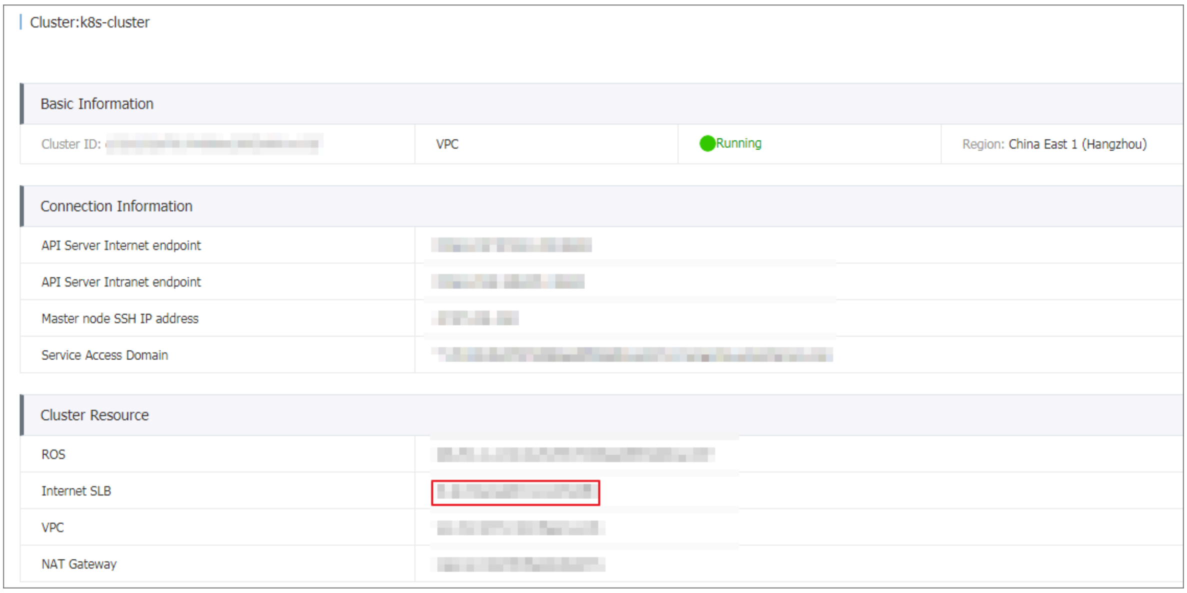
After you run the command, the following message is displayed, indicating that you have connected to the cluster.

1.4.9 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).

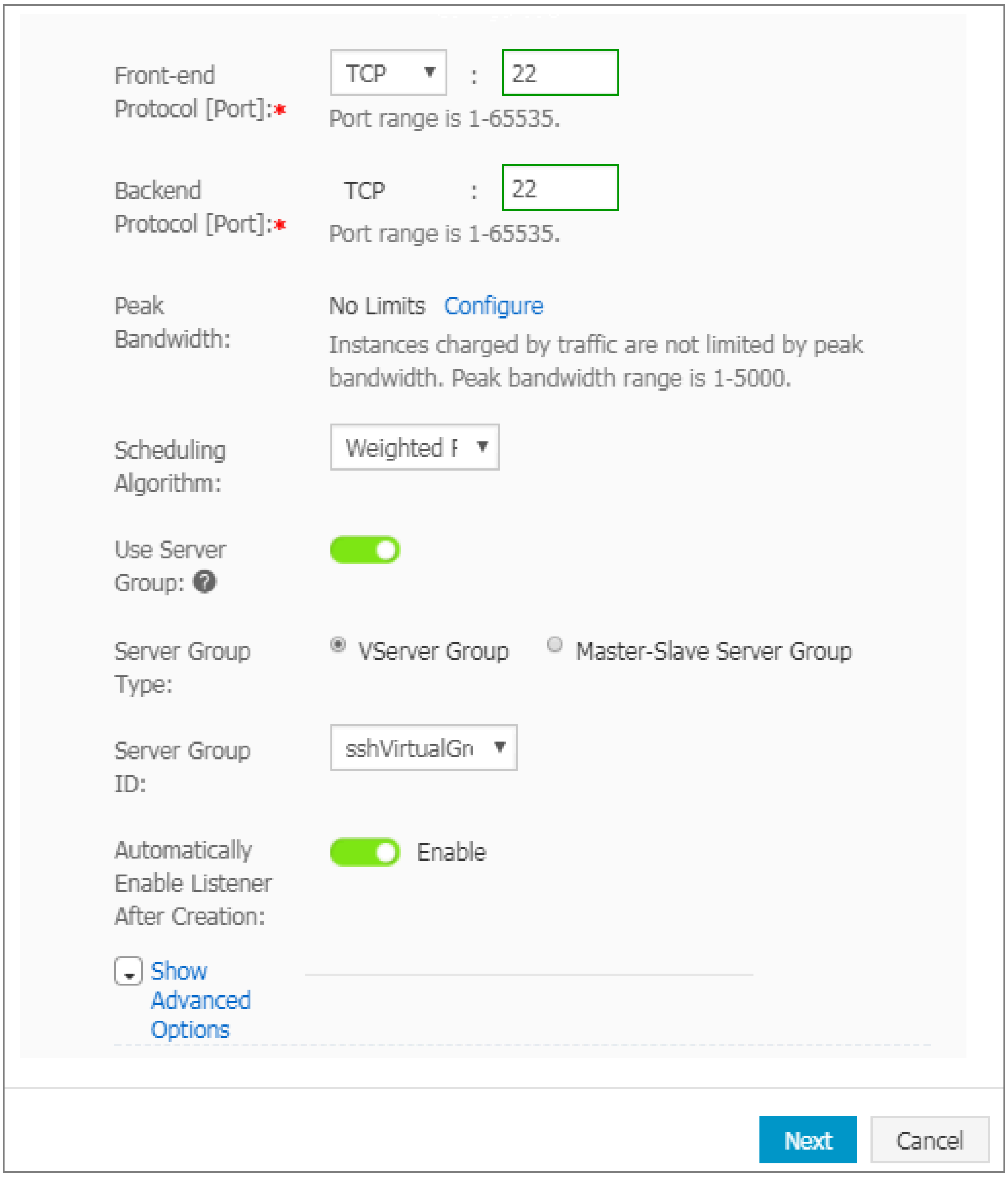
Procedure

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.

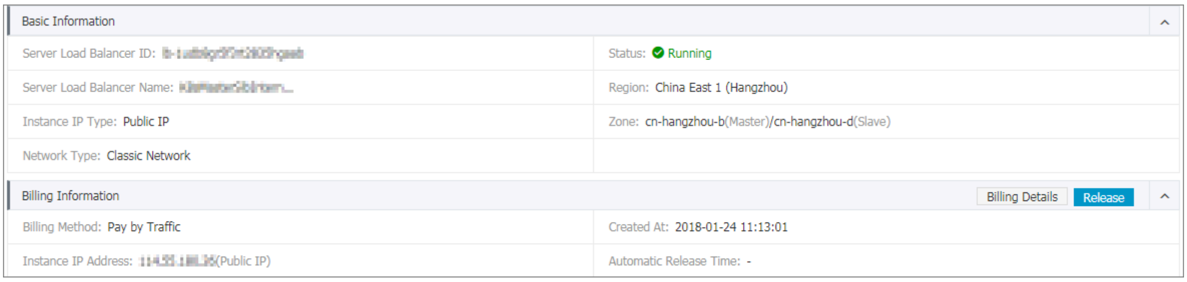


5. Select Instances > Server Load Balancer, and click Add Listener.
6. Add the SSH listening rule.

   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.



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



1.4.10 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_40*. Then, click Create.
   a. 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.
   b. 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`.

1.4.11 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.

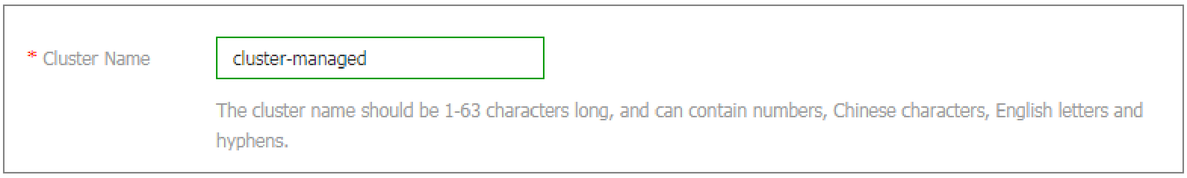
Context

- 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.

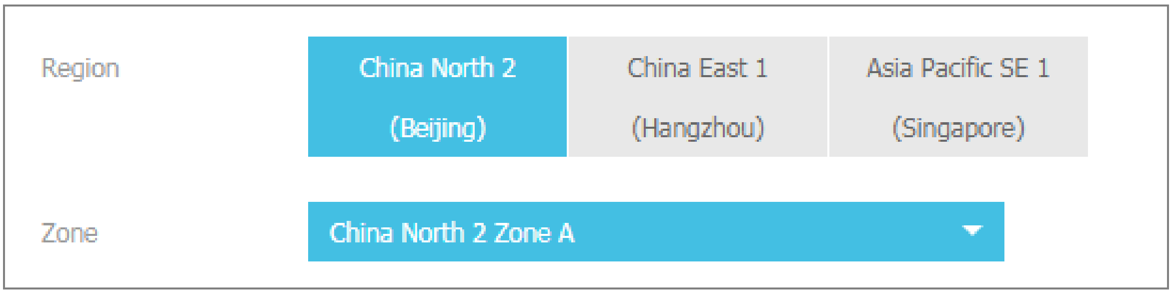
Procedure

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 (-).



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



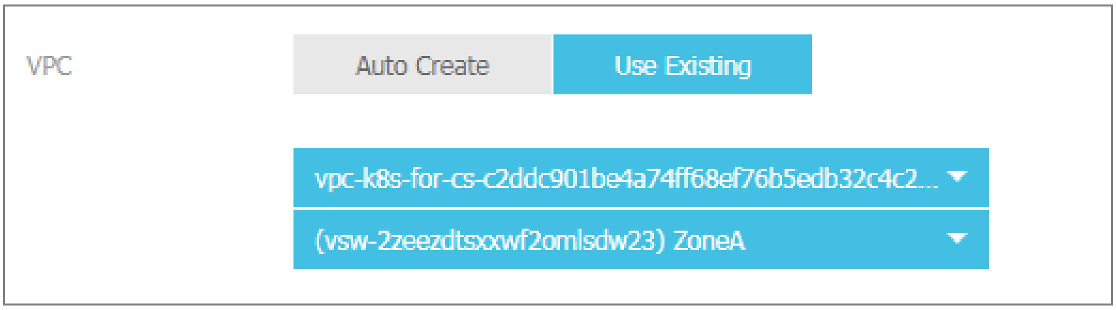
6. Set the cluster network type.

> 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 selectUse 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.



7. Set the node type.

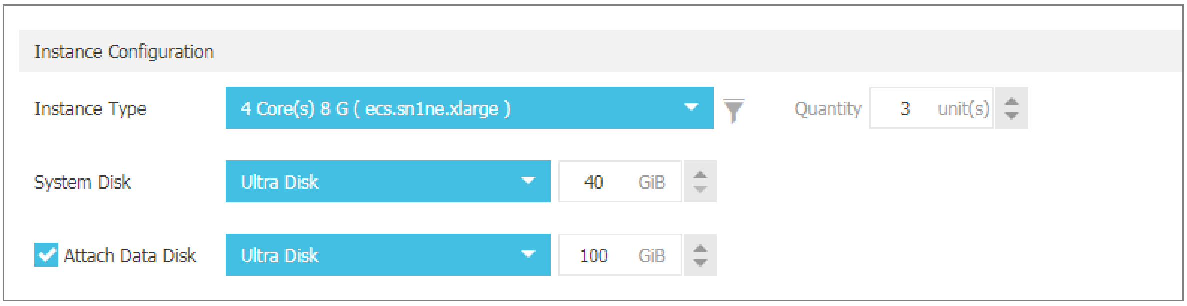
Note:

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

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.

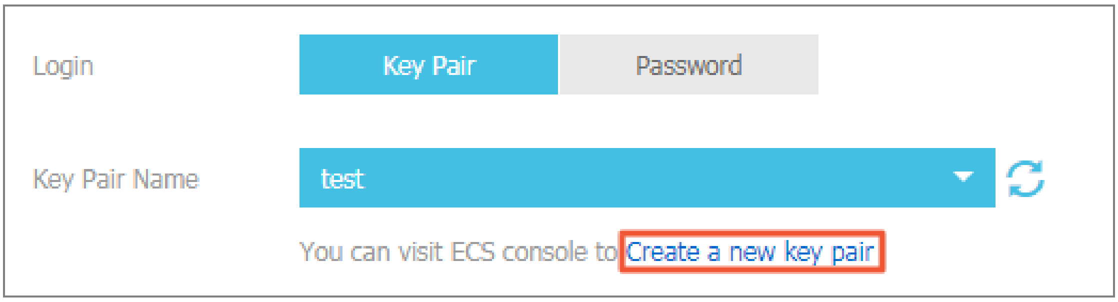


9. Set the logon mode.

- 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.

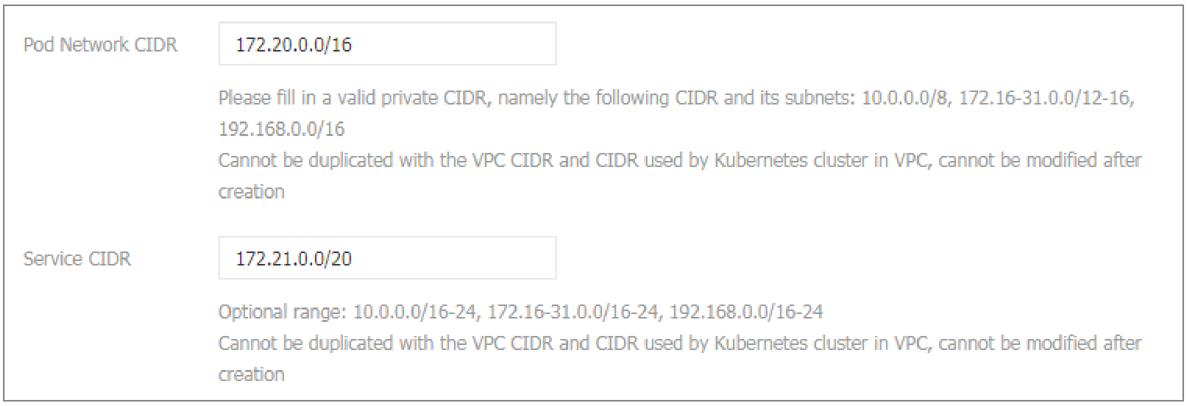
- Set the password.
  - Logon Password: Set the node logon password.
  - Confirm Password: Confirm your node logon password.



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

Note:

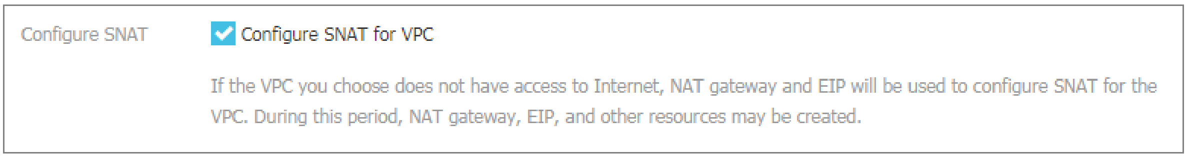
- These two parameters are available only when you select to Use Existing VPC.
- Both Pod Network CIDR and Service CIDR cannot overlap with the Classless Inter-Domain Routing (CIDR) block used by the VPC and the existing Kubernetes clusters in the VPC. The values cannot be modified after the cluster 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*.



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

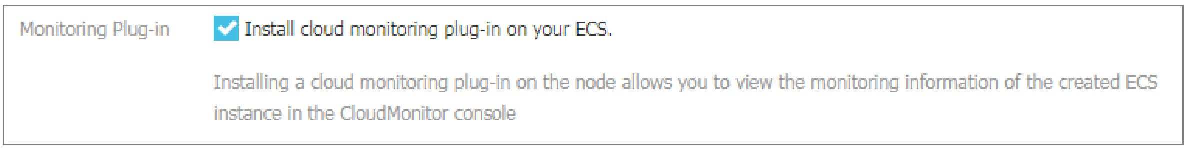
Note:

- 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.



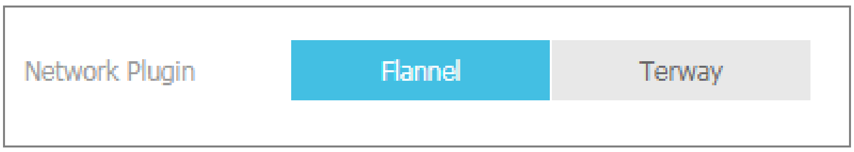
12. Select whether to install a cloud monitoring plug-in on your ECS.

    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.



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

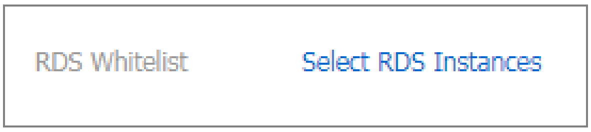


14. Set the RDS whitelist.

    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.



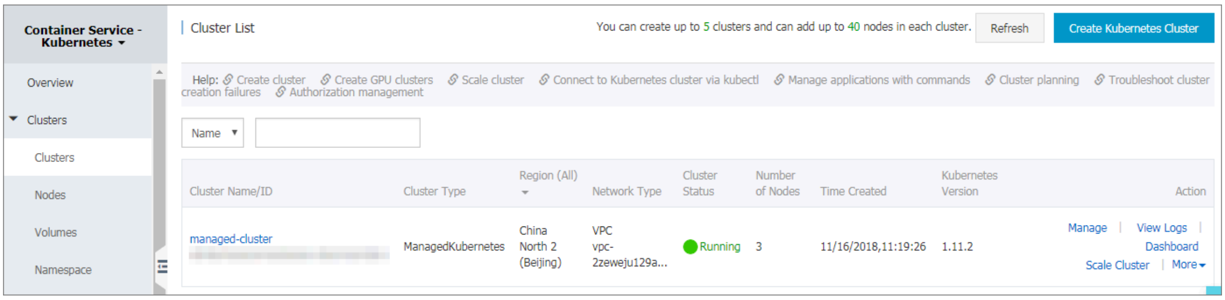
15.Click Create in the upper-right corner.

Note:

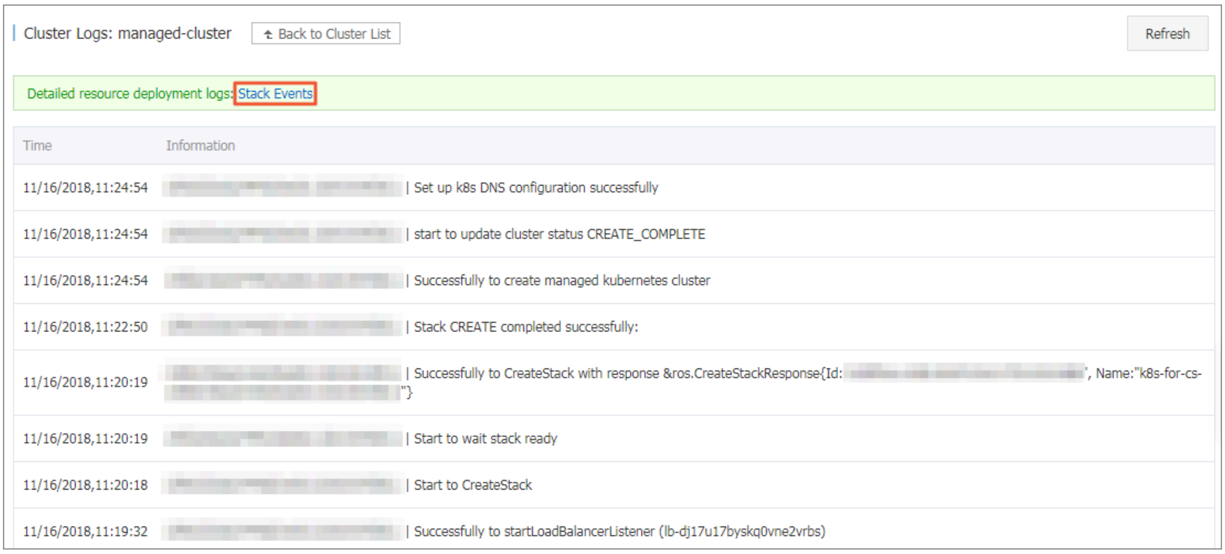
Normally, it takes five minutes to create a multi-node Kubernetes cluster.

Result

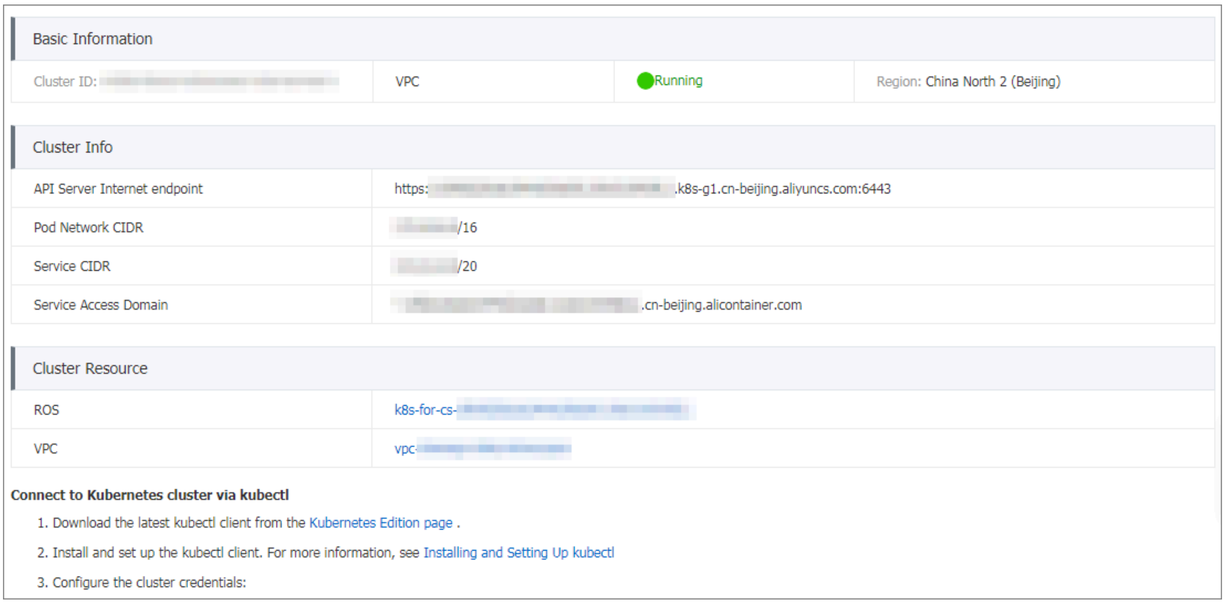
After the cluster is successfully created, you can view the cluster on the Cluster List page of the Container Service console.



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.



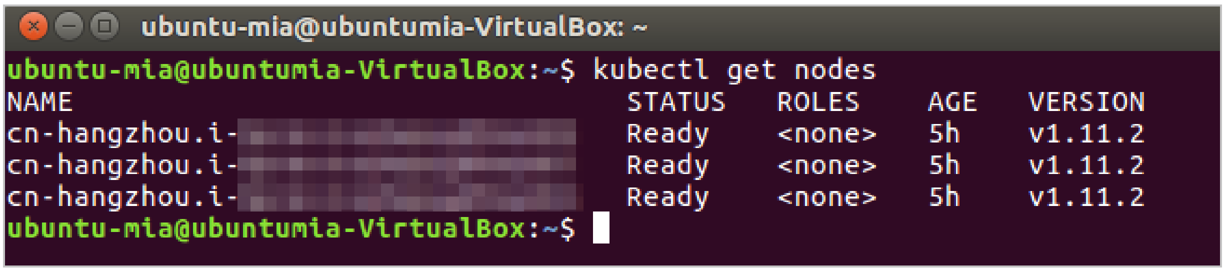
On the Cluster List page, find the created cluster and click Manage to view the basic information and connection information about this cluster.



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`.

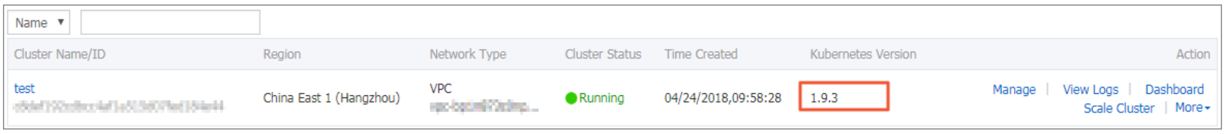
You can see *Connect to a Kubernetes cluster by using kubectl* and run `kubectl get node` to view the node information of the cluster.



1.4.12 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.



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.

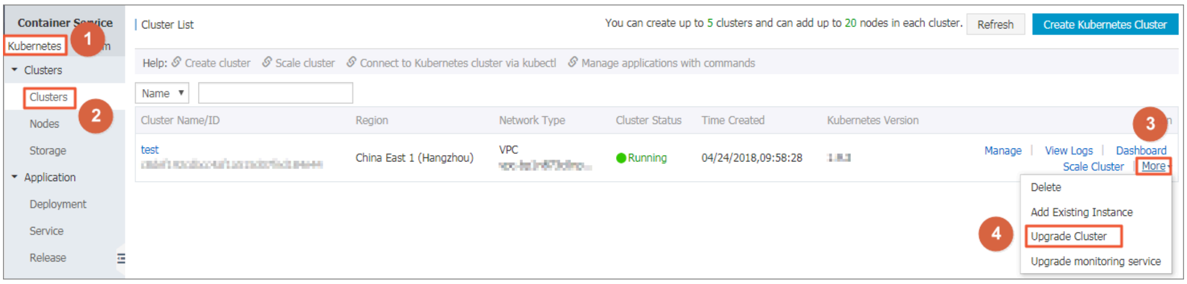
```
kubectl get nodes
 NAME                     STATUS     ROLES      AGE        VERSION
 cn-shanghai.i-xxxxxx     Ready      master     38d        v1.9.3
```

```
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.



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.

1.4.13 Upgrade a system component

This topic describes how to upgrade a system component.

Prerequisites

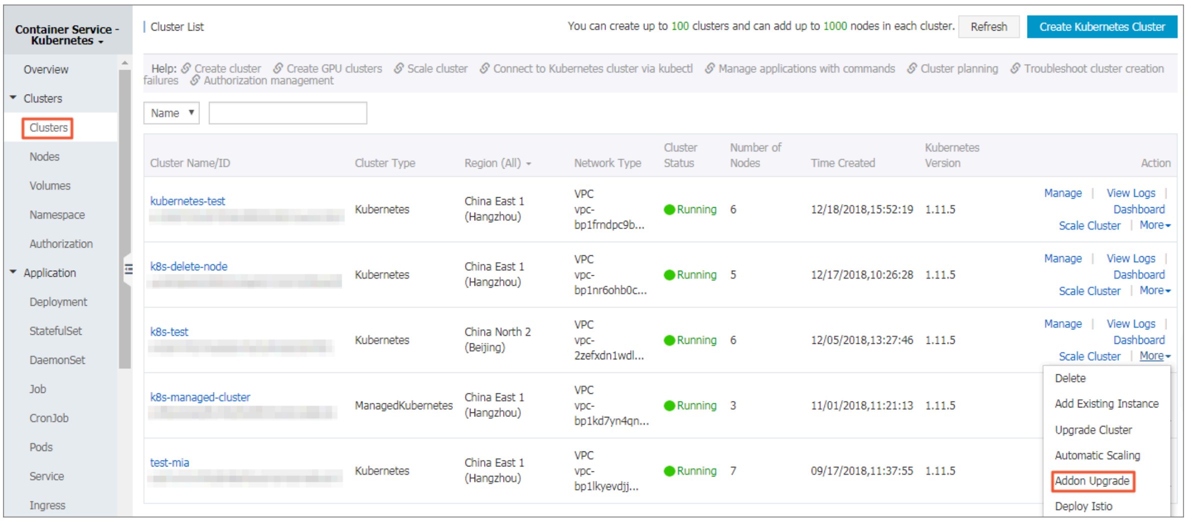
You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

Context

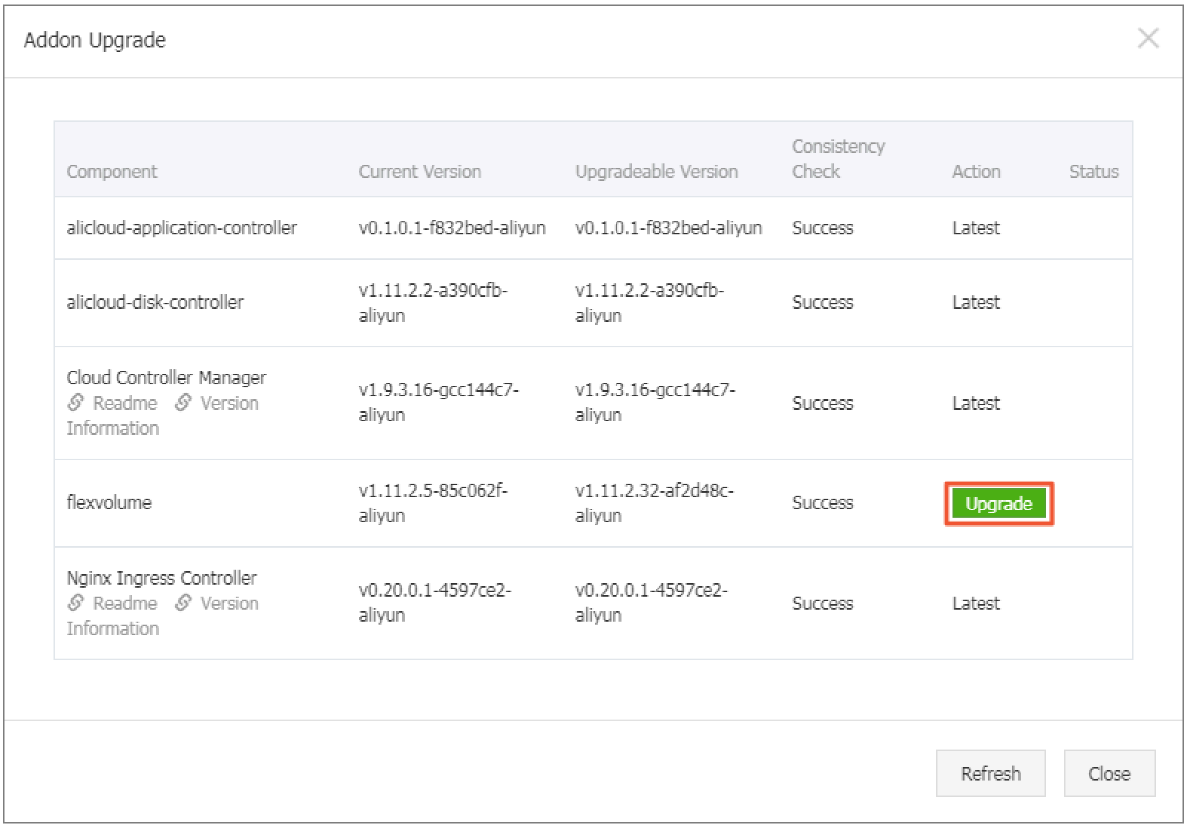
The following procedure is for if you need to independently upgrade one or multiple system components of a Kubernetes cluster even if the cluster is of the latest version.

Procedure

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

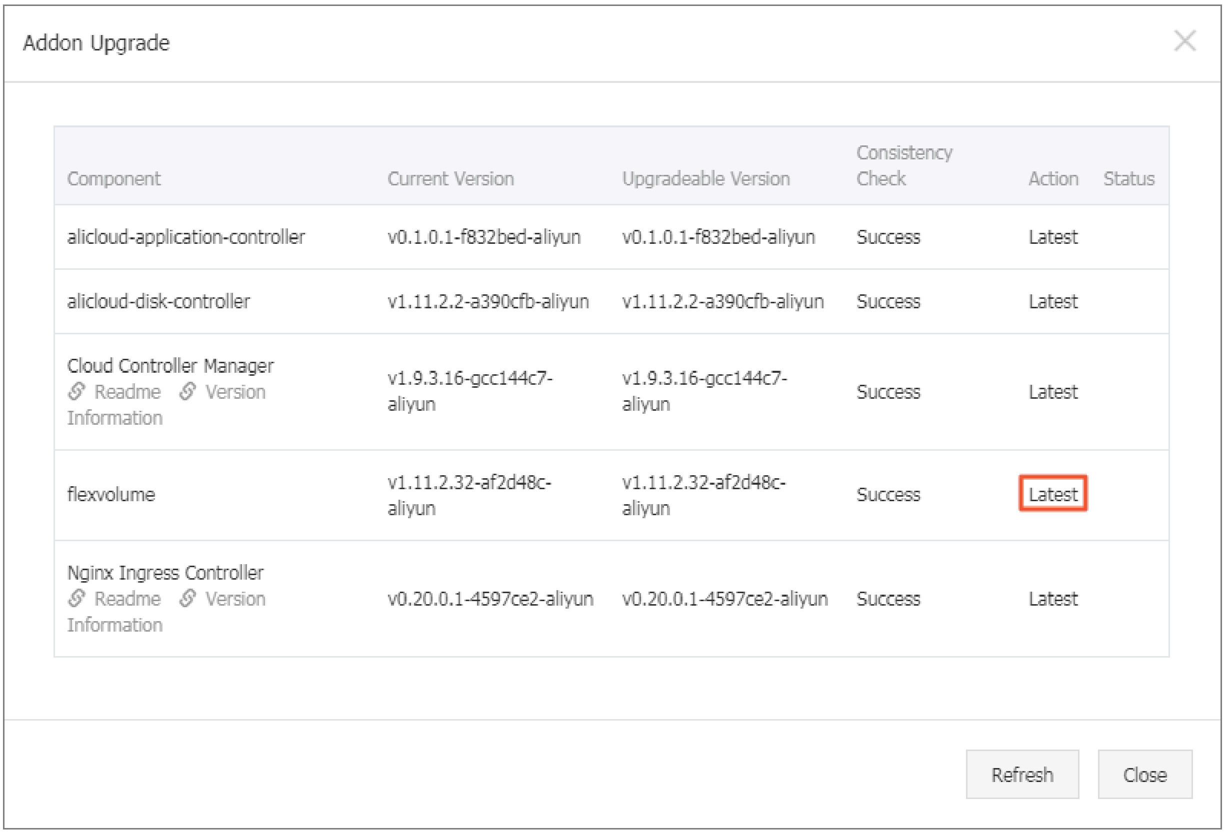


4. Select the target system component, and click Upgrade in the Action column. Upgrading is then displayed in the Status column.



Result

On the Addon Upgrade page, Latest is displayed in the Action column of the target system component.



Component	Current Version	Upgradeable Version	Consistency Check	Action	Status
alicloud-application-controller	v0.1.0.1-f832bed-aliyun	v0.1.0.1-f832bed-aliyun	Success	Latest	
alicloud-disk-controller	v1.11.2.2-a390cfb-aliyun	v1.11.2.2-a390cfb-aliyun	Success	Latest	
Cloud Controller Manager Readme Version Information	v1.9.3.16-gcc144c7-aliyun	v1.9.3.16-gcc144c7-aliyun	Success	Latest	
flexvolume	v1.11.2.32-af2d48c-aliyun	v1.11.2.32-af2d48c-aliyun	Success	Latest	
Nginx Ingress Controller Readme Version Information	v0.20.0.1-4597ce2-aliyun	v0.20.0.1-4597ce2-aliyun	Success	Latest	

1.4.14 Update the Kubernetes cluster certificates that are about to expire

This topic describes how to update the Kubernetes cluster certificates that are about to expire through the Container Service console.

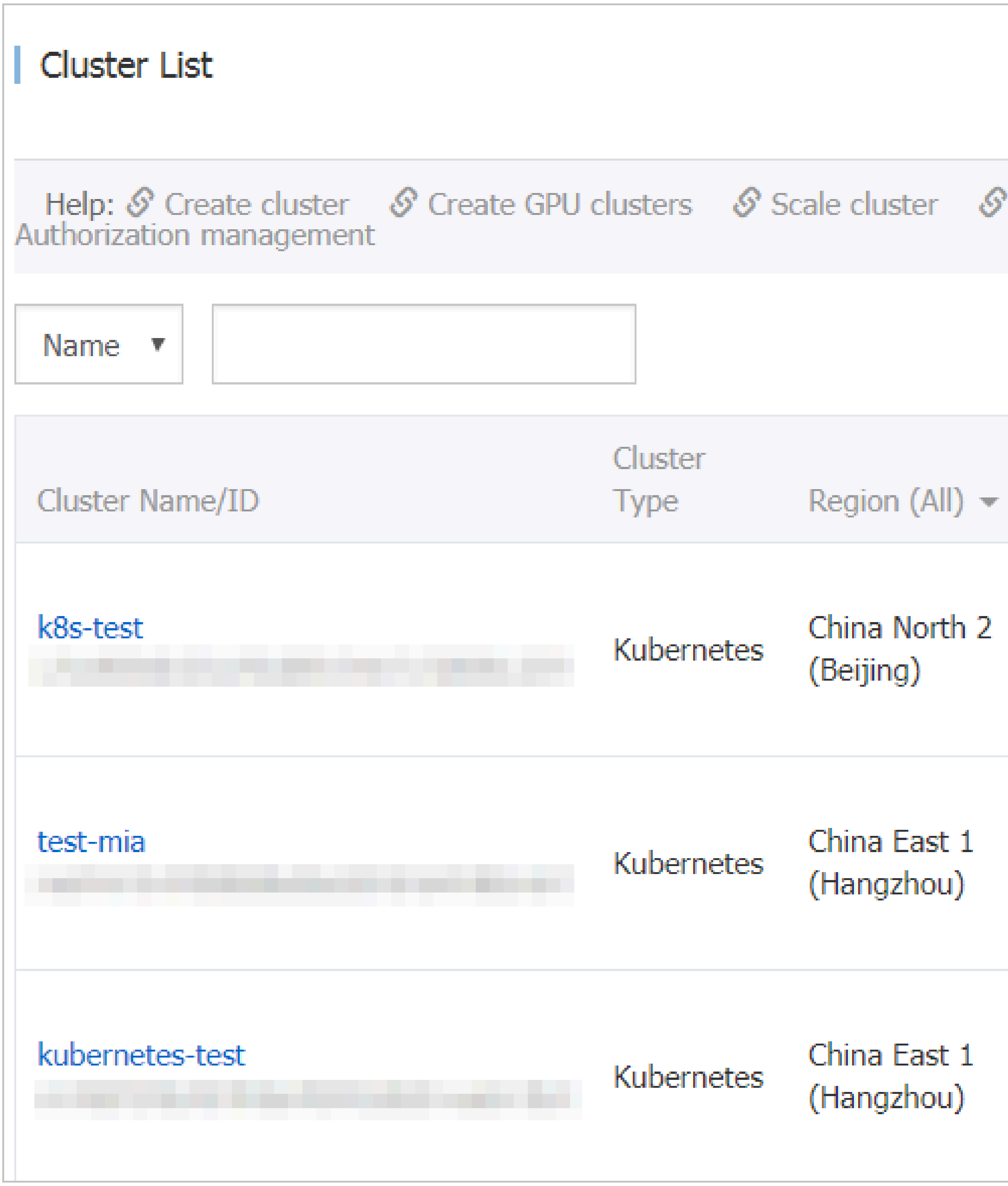
Prerequisites

You have created a Kubernetes cluster and the system has already prompted you to update the cluster certificates. For more information, see *Create a Kubernetes cluster*.

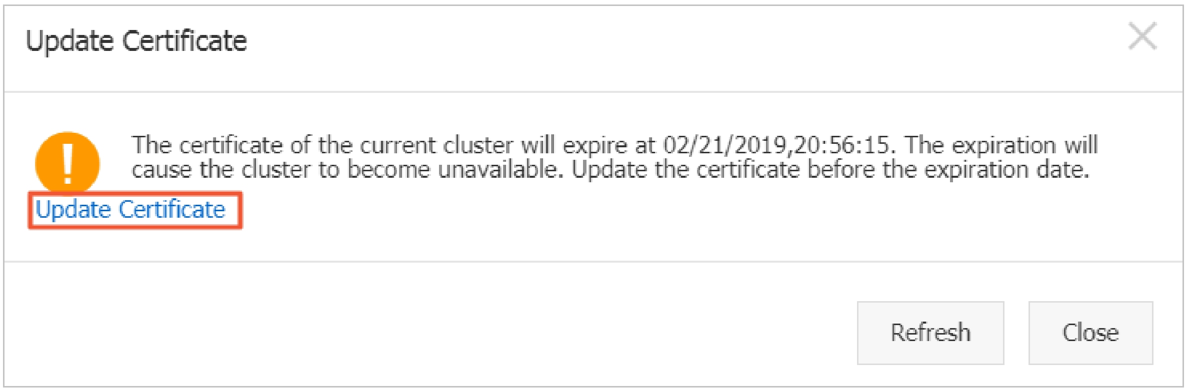
Procedure

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

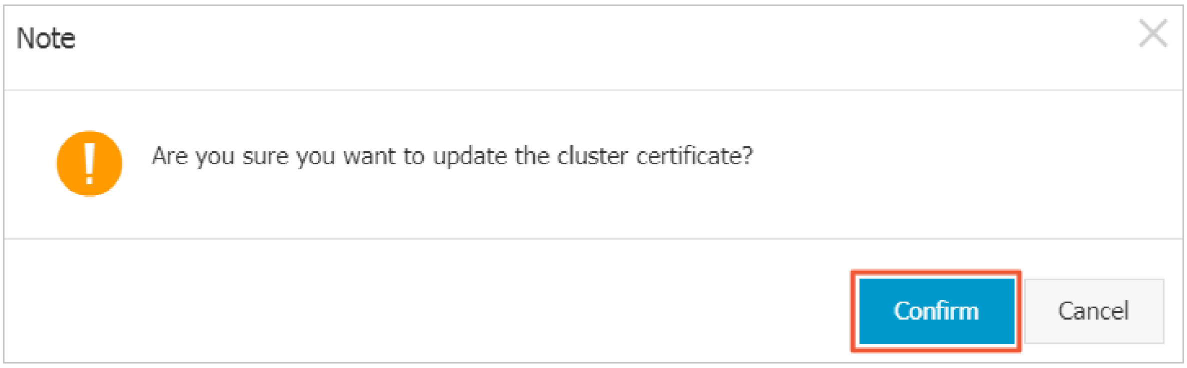
> **Note:**
> If cluster certificates are about to expire in about two months, the system displays the Update Certificate prompt for the cluster.



4. Click Update Certificate.

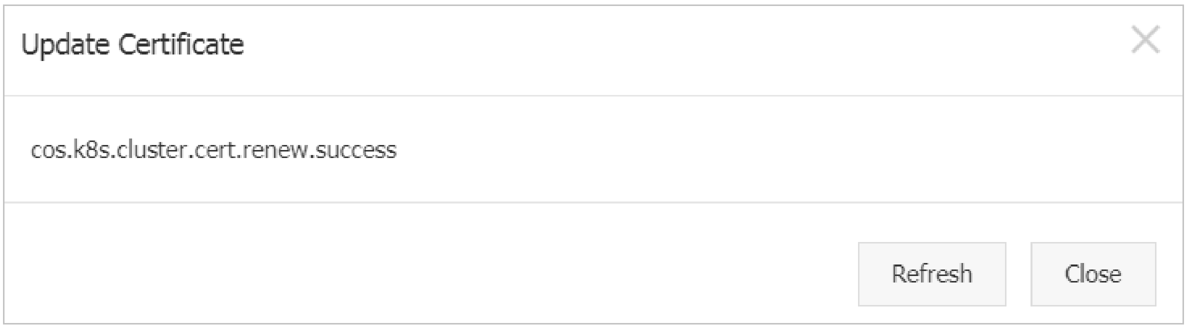


5. Click Confirm.



Result

- The Update Certificate page displays Success.



- On the Cluster List page, the Update Certificate prompt of the target cluster has been removed.

1.4.15 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.

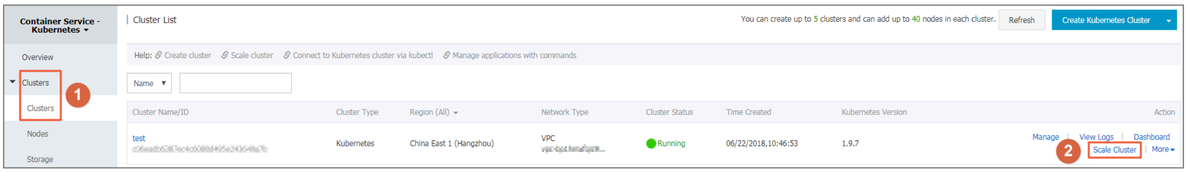
Context

- 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.

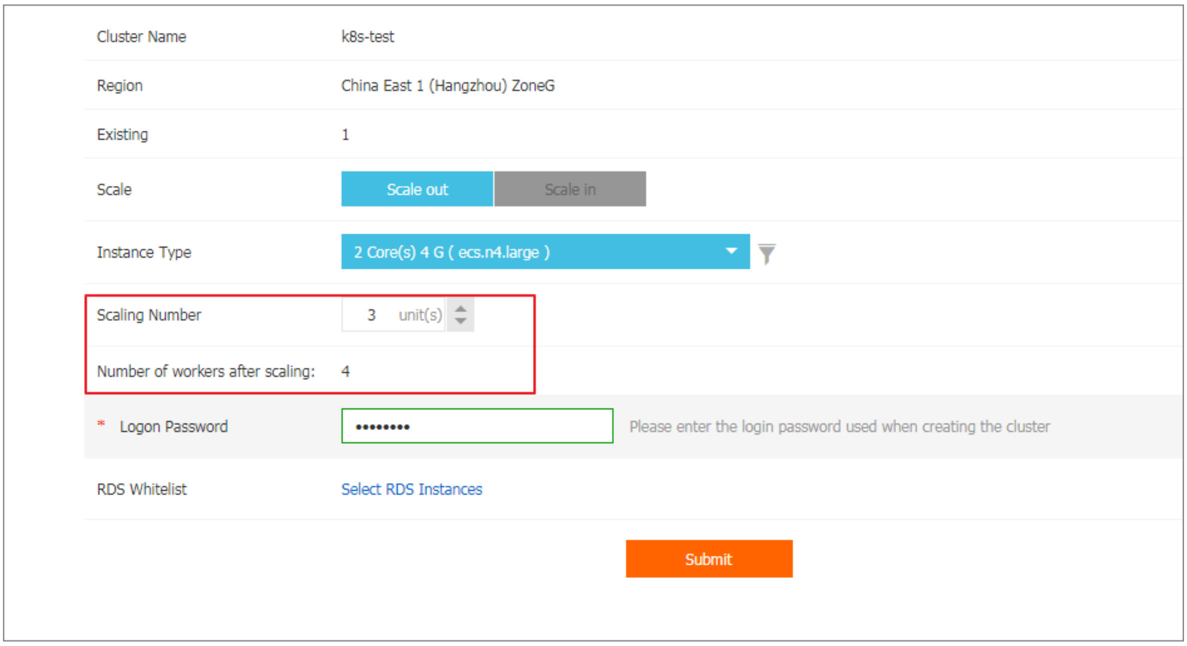
Procedure

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.



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.



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.

1.4.16 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*.

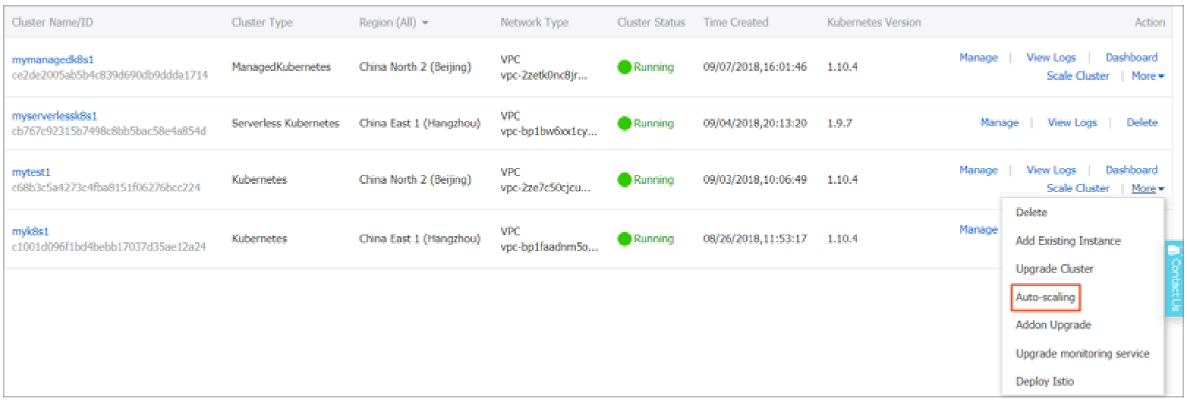
Background

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

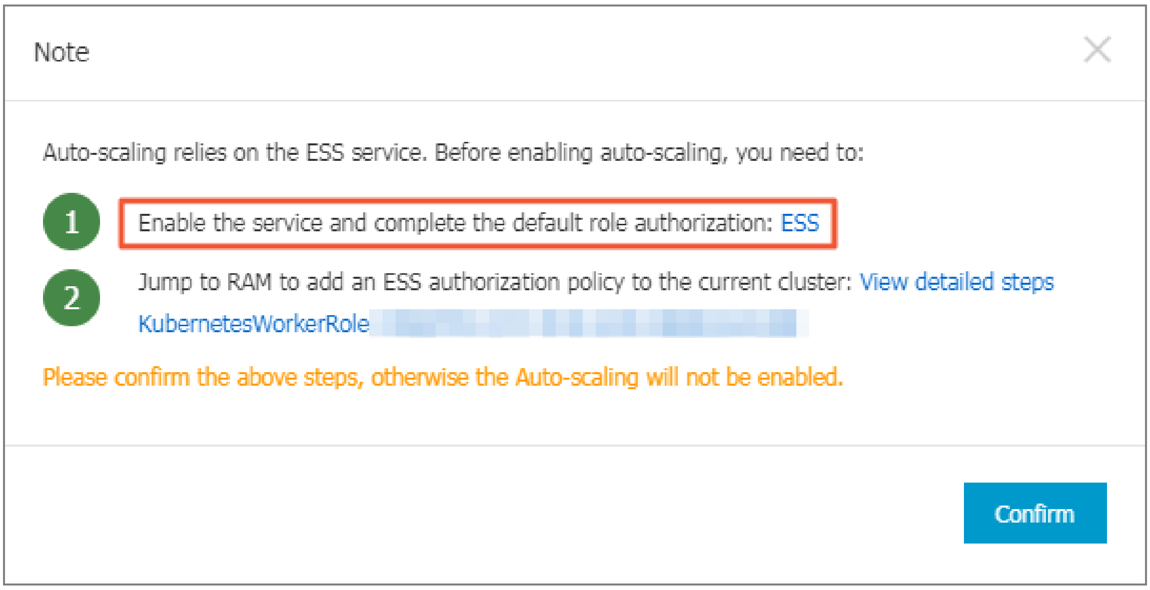
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Clusters in the left-side navigation pane.
3. Select a cluster and click More > Auto Scaling in the Action column.



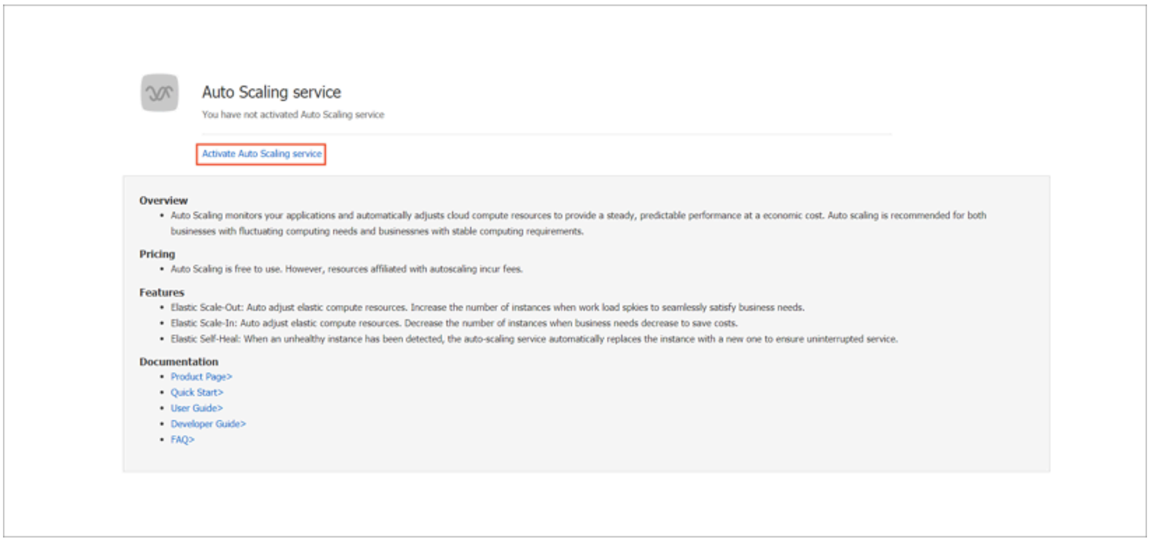
Authorization

- Activate Auto Scaling service

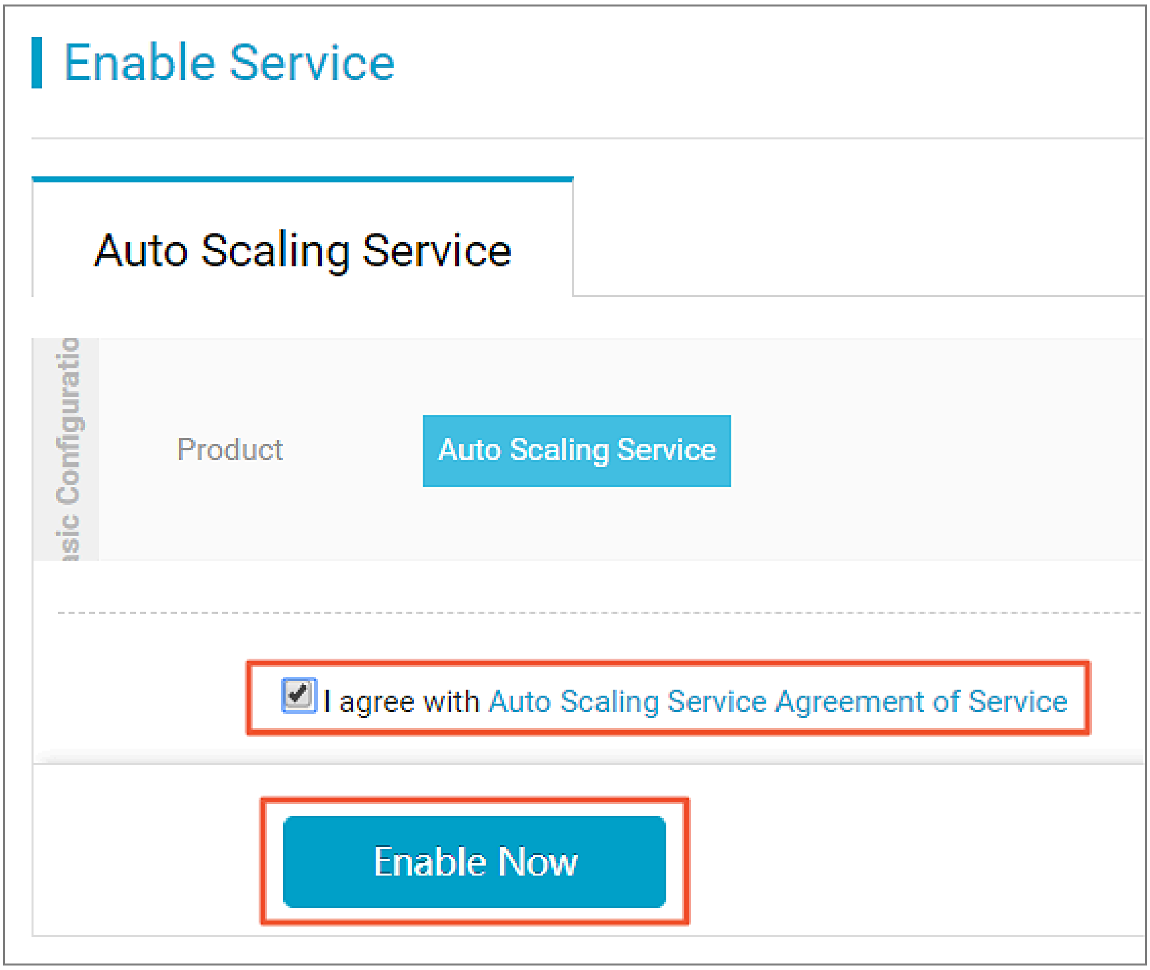
1. Click the first link in the displayed dialog box.



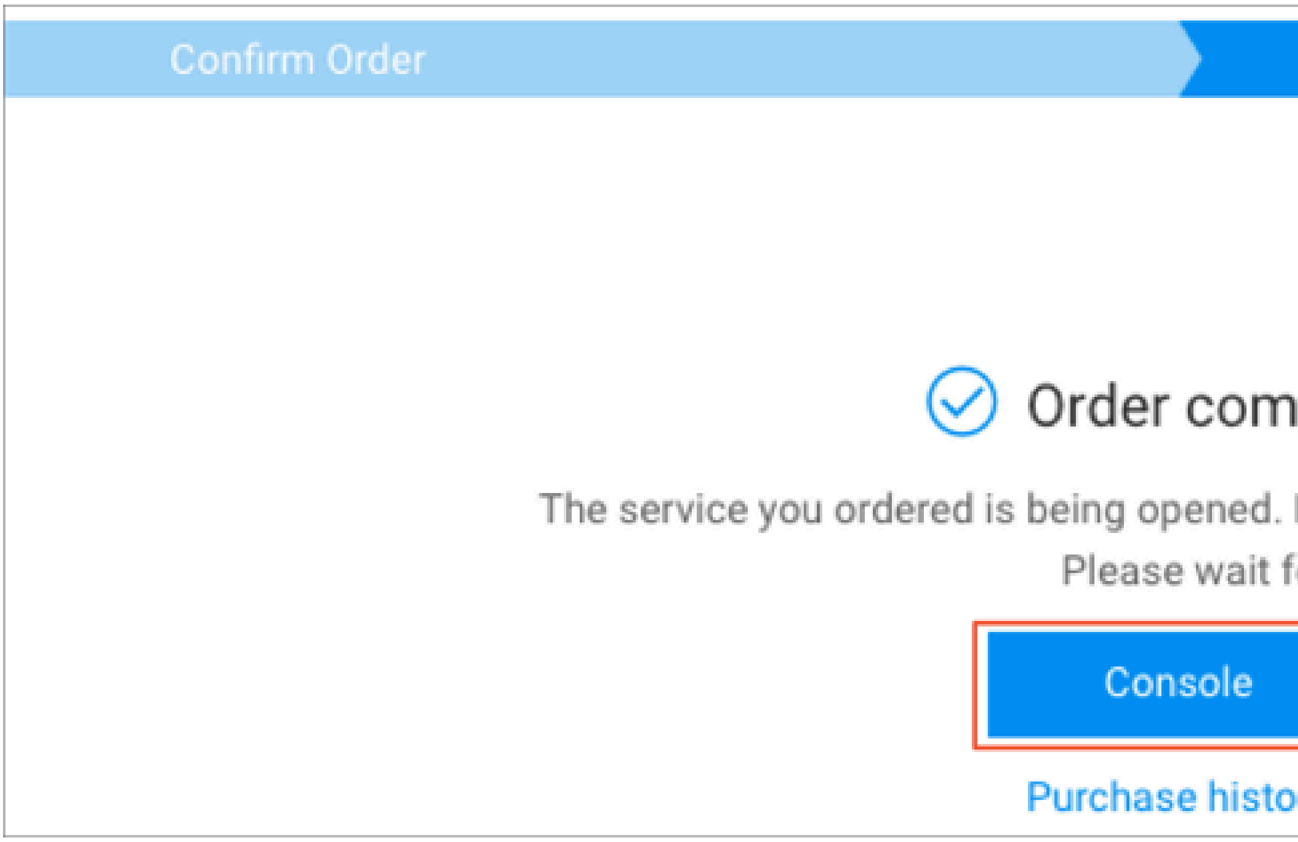
2. Click Activate Auto Scaling service.



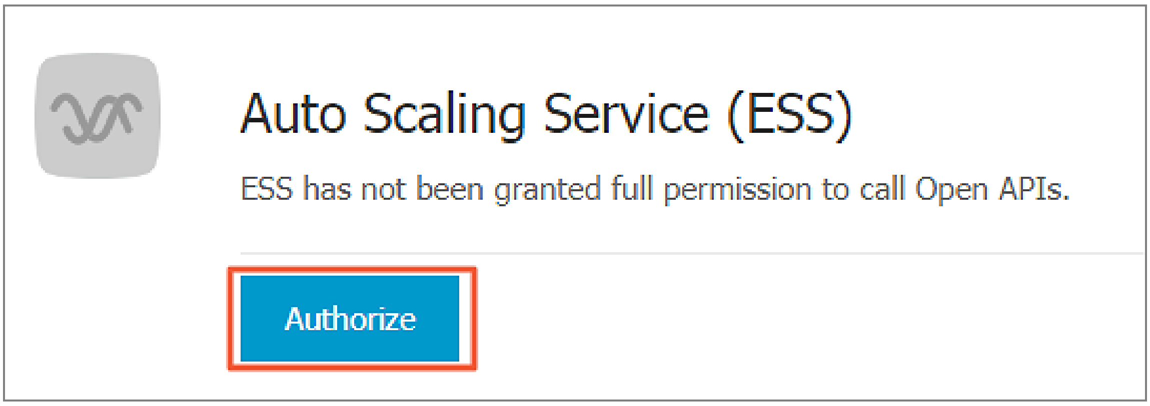
3. Select the I agree with Auto Scaling Service Agreement of Service check box and click Enable Now.



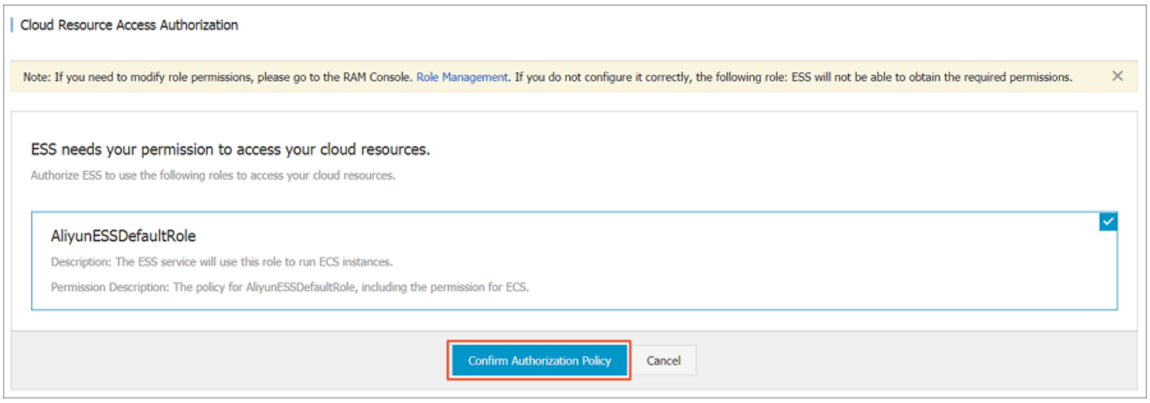
4. After the service is activated, click Console.



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



6. Click Confirm Authorization Policy.



Expected result

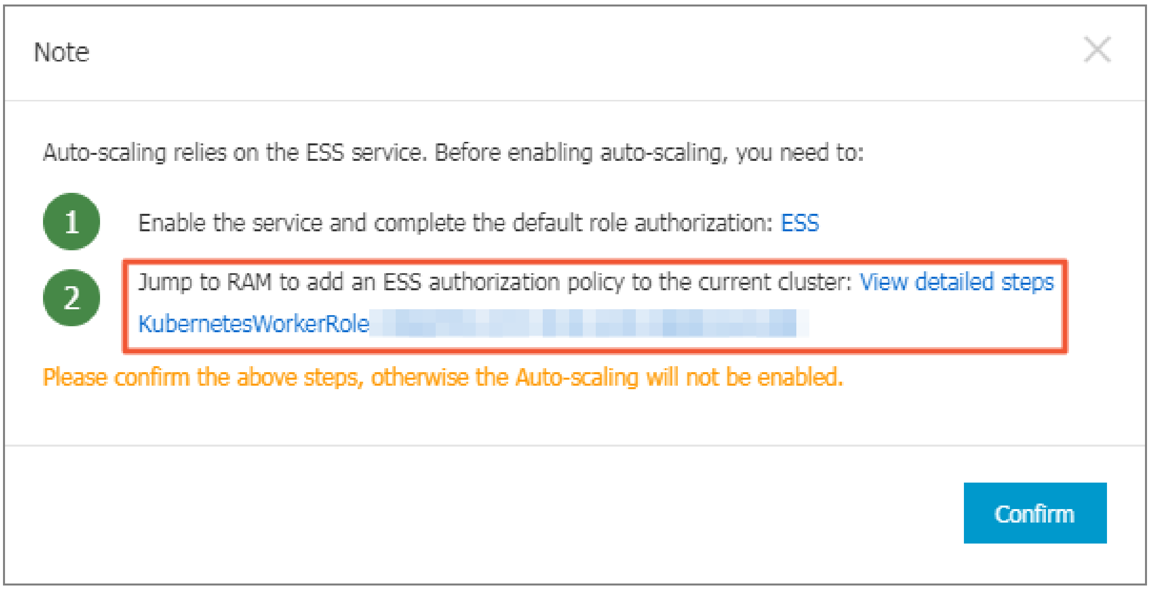
When the page automatically jumps to the elastic scaling console, the authorization succeeds. Closes the page and continues to configure Authorize roles

.

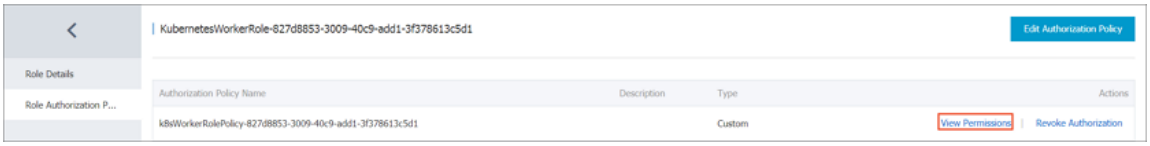
- Authorize a role.

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

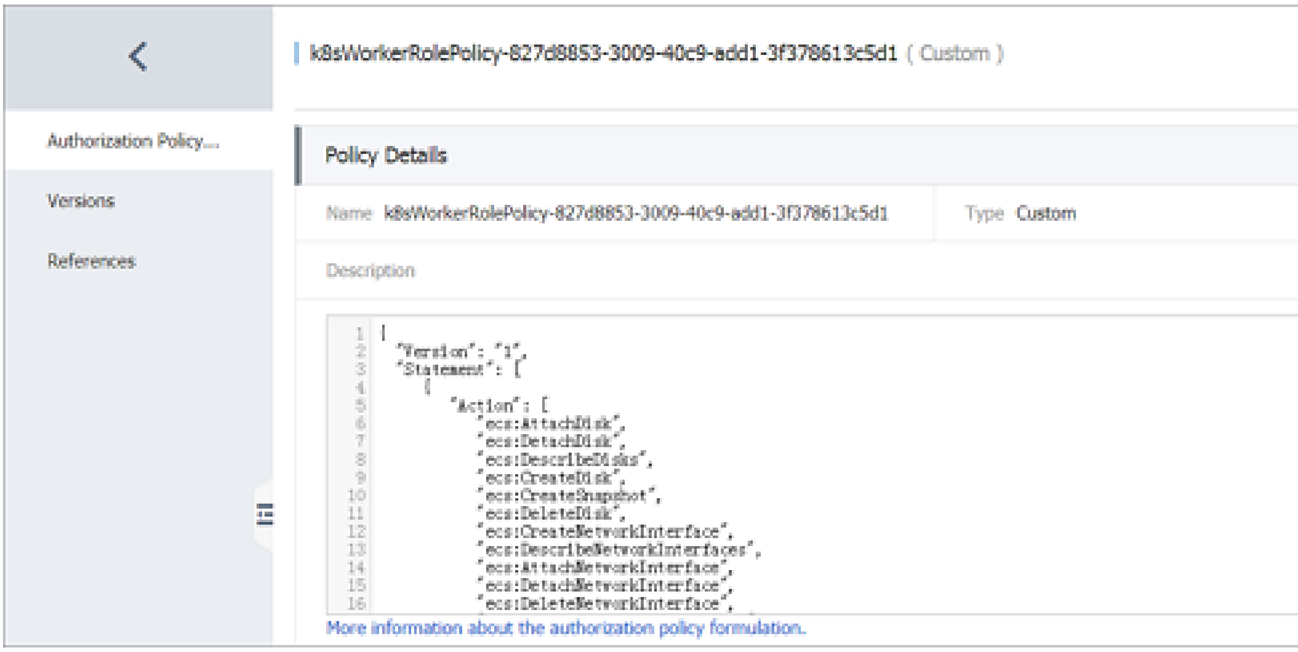
> **Note:**
> Use the primary account to log on to the console.



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



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



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.

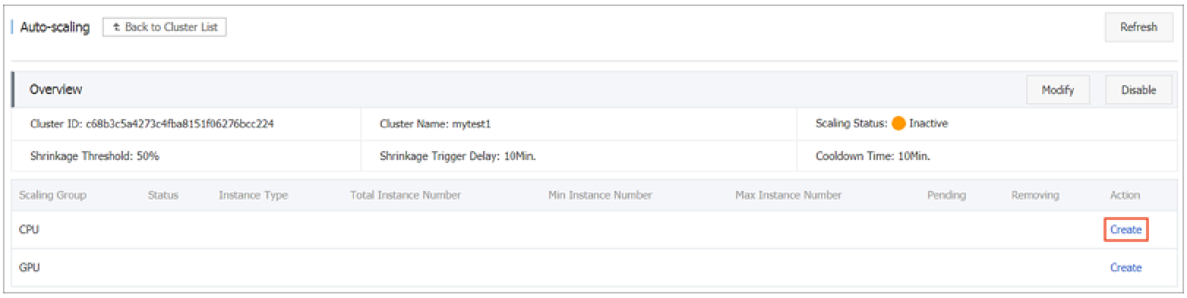
5. Click Modify Authorization.

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 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.



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

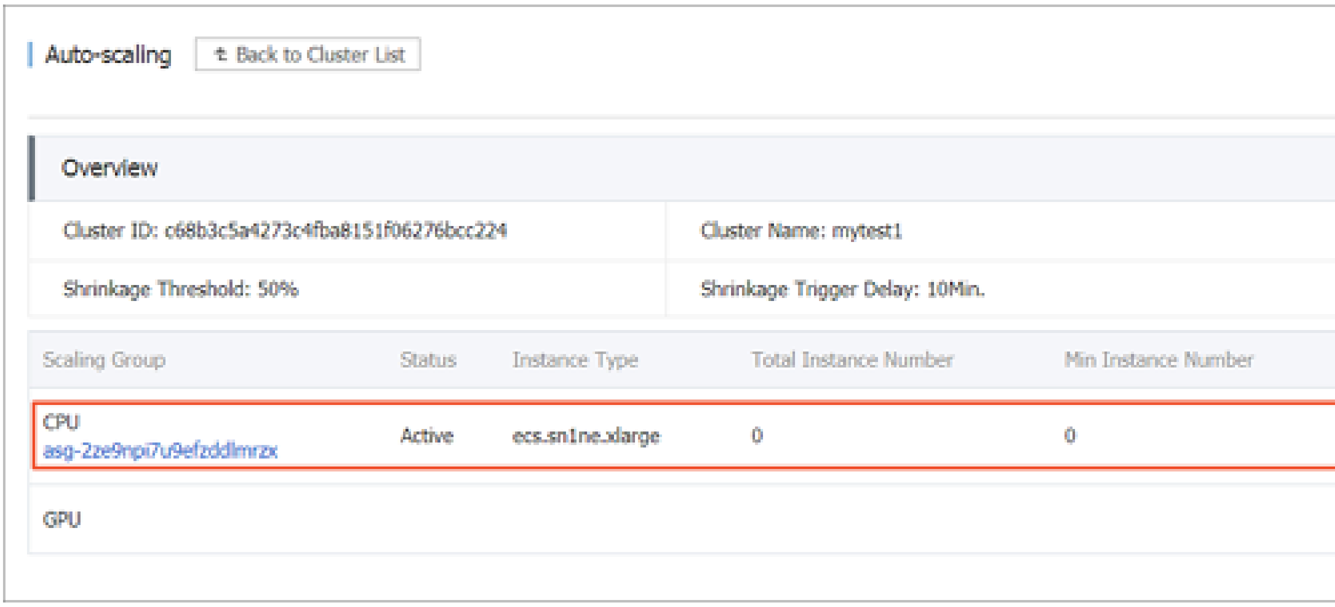
Configuration	Description
Instance Type	Types of instances in the scaling group.
System Disk	The system disk of the scaling group.
Attach Data Disk	Whether or not to mount a data disk when you create a scaling group. By default, no data disk is not mounted.
Instance Quantity	The number of instances contained 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.

Configuration	Description
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.

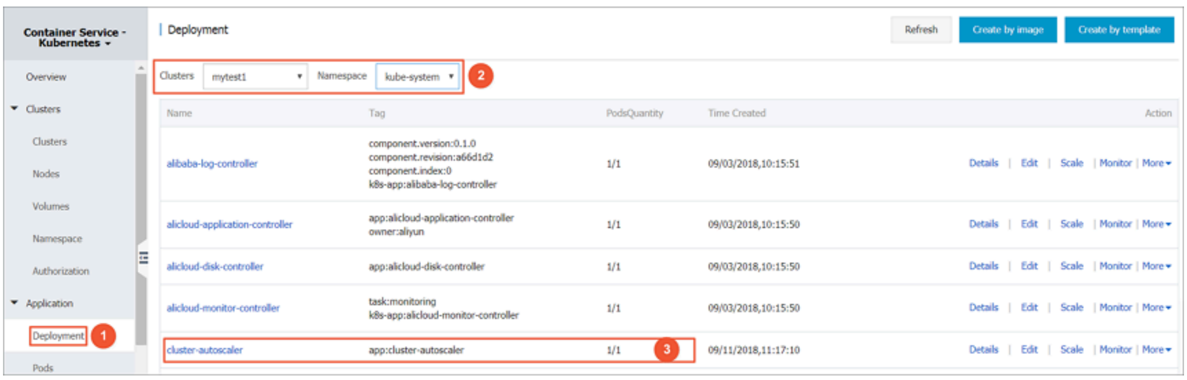
3. Click OK to create a scaling group.

Expected result

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



- 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.



1.4.17 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.

ClickMore > 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_52*.

1.5 Node management

1.5.1 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

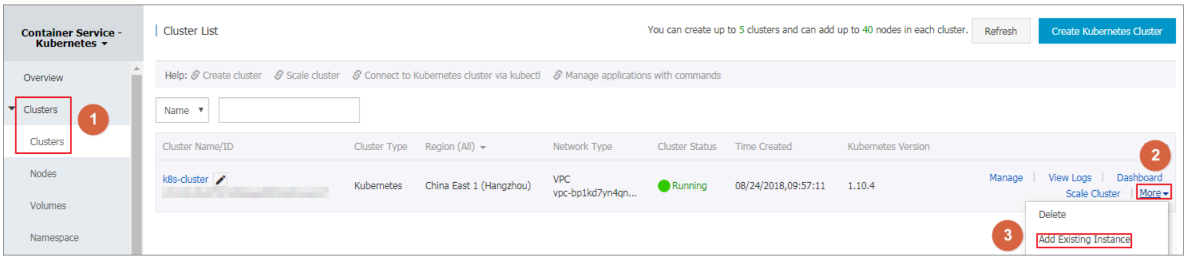
- By default, each cluster can contain up to 40 nodes. To add more nodes, open a ticket.
- The ECS instance to be added must be in the same Virtual Private Cloud (VPC) region as the cluster.
- When adding an existing instance, make sure that your instance has an Elastic IP (EIP) for the VPC network type, 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.

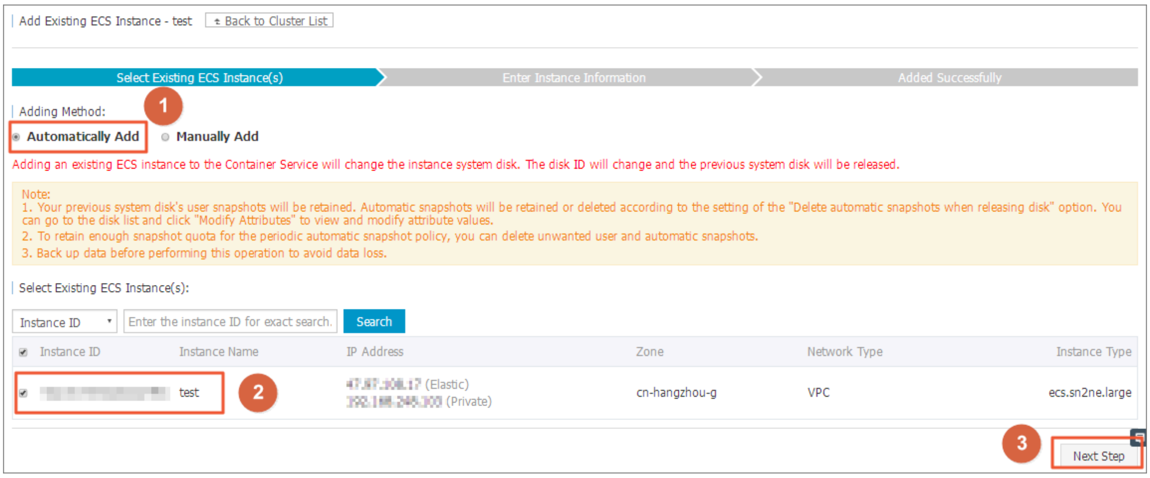
   The Add Existing ECS Instance page appears. All the available ECS instances under the current account are displayed on this page. Select to add existing ECS instances automatically or manually.

   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.

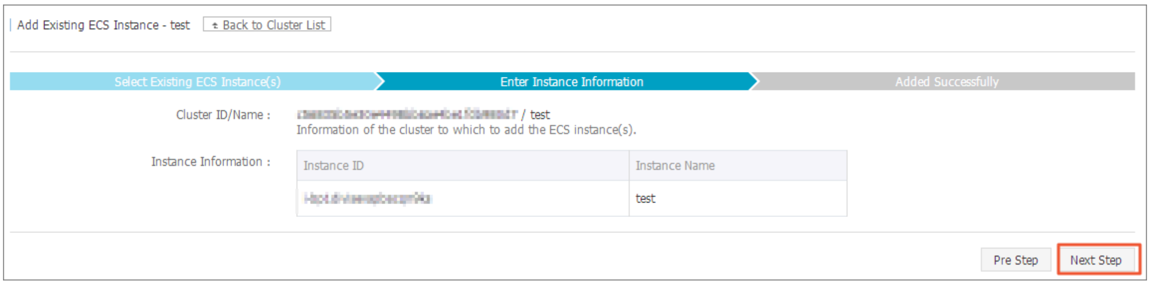


4. Select Automatically Add to add multiple ECS instances at a time.

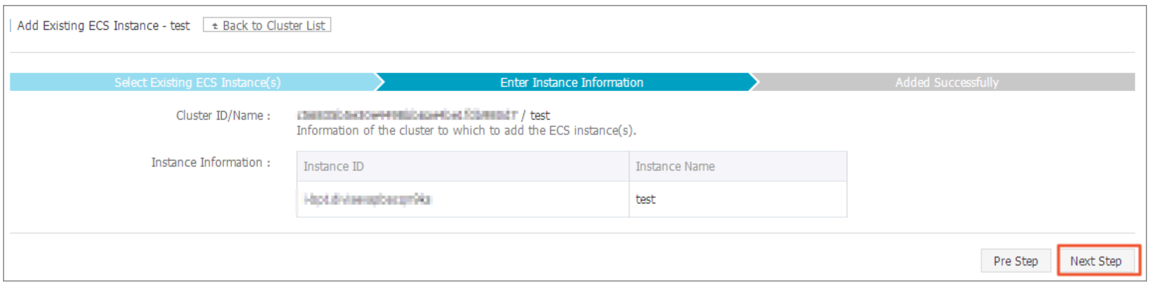
a) In the list of existing cloud servers, select the target ECS instance, and then click Next Step.



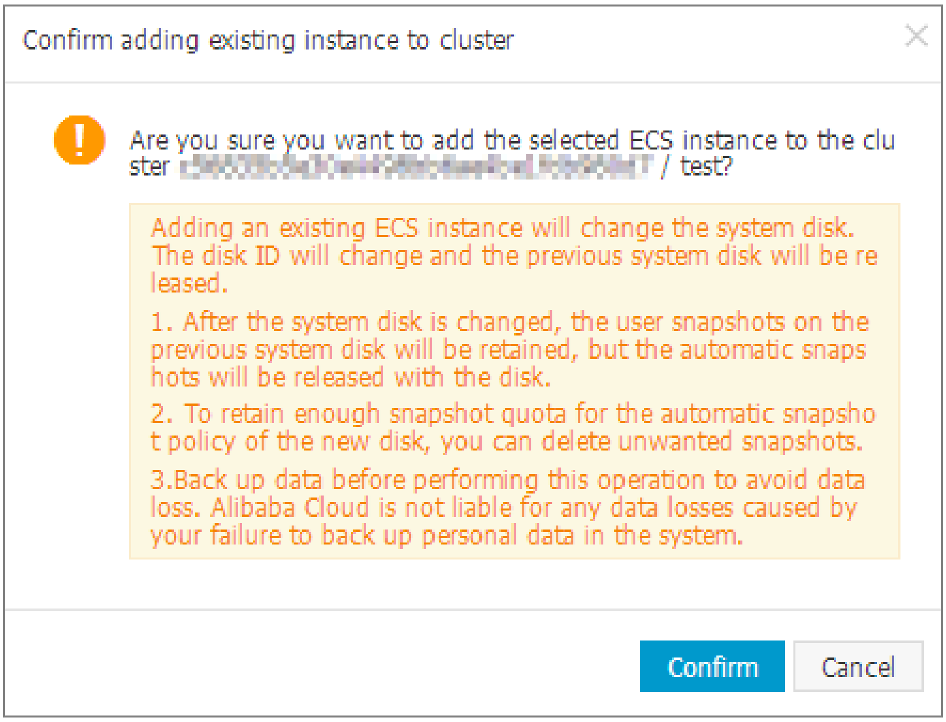
b) Enter the instance information, set the logon password, and then click Next Step.



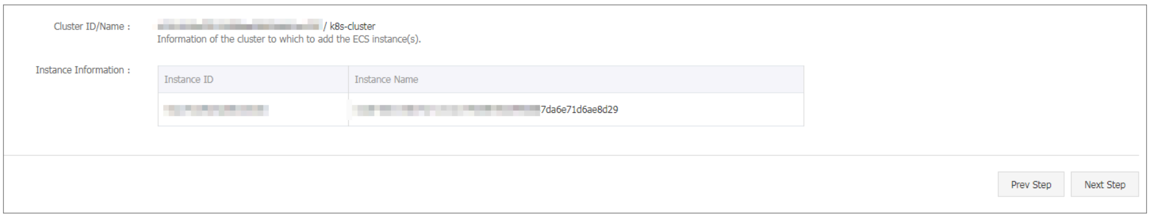
c) Click Confirm in the displayed dialog box. The selected ECS instances are automatically added to this cluster.



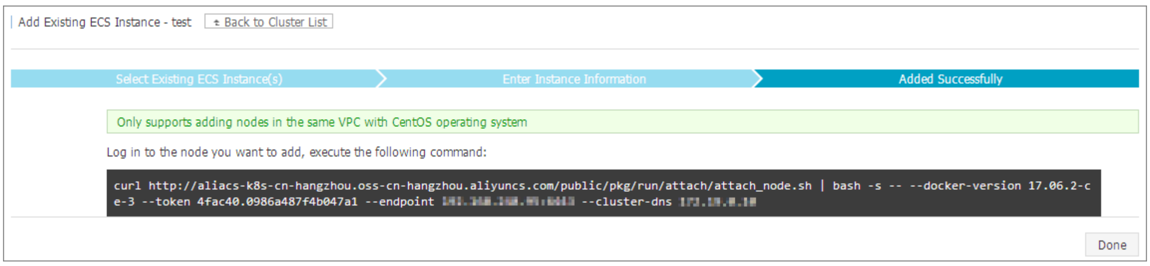
5. Optional: You can also select Manually Add to manually add an existing ECS instance to the cluster.
    a) Select the ECS instance to be added and then click Next Step. You can add only one ECS instance at a time.



    b) Confirm the information and then click Next Step.

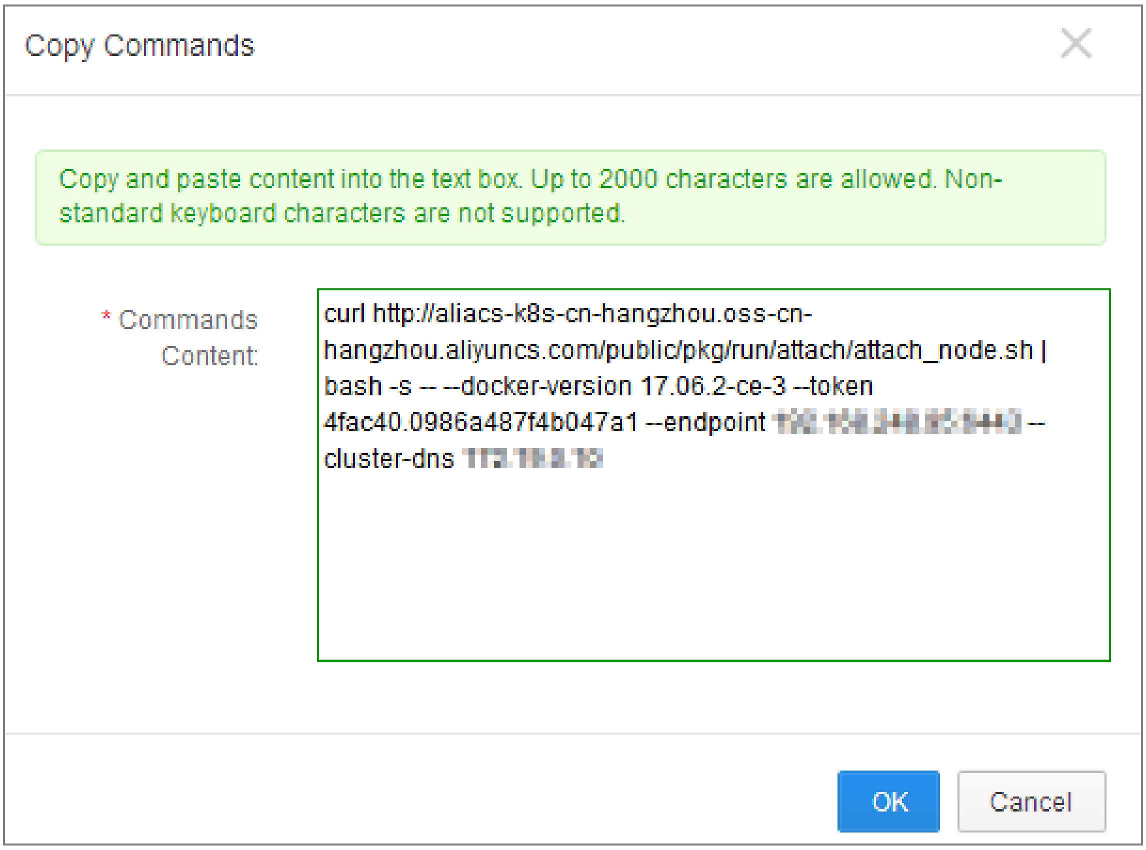


    c) Copy the command.



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

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



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

1.5.2 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_56* 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 > > Nodesin the left-side navigation pane.
3. Select the cluster from the Cluster drop-down list and then view the node list of this cluster.

View node list in Kubernetes dashboard

1. Log on to the *Container Service console*.
2. Click Kubernetes > Clusters in the left-side navigation pane.
3. Click Dashboard at the right of the cluster to enter the Kubernetes dashboard.
4. In the Kubernetes dashboard, click Nodes in the left-side navigation pane to view the node list of this cluster.

1.5.3 Node monitoring

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

Procedure

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

4. Click Monitor at the right of the node to view the monitoring information of this node.

5. You are redirected to the CloudMonitor console. View the basic monitoring information of the corresponding ECS instance, including the CPU usage, network inbound bandwidth, network outbound bandwidth, disk BPS, and disk IOPS.

What's next

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

Kubernetes clusters can now monitor resources by using application groups. For more information, see *#unique_59*.

1.5.4 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_61*.

Prerequisite

You have successfully created a Kubernetes cluster. For more information, see *#unique_40*.

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`.

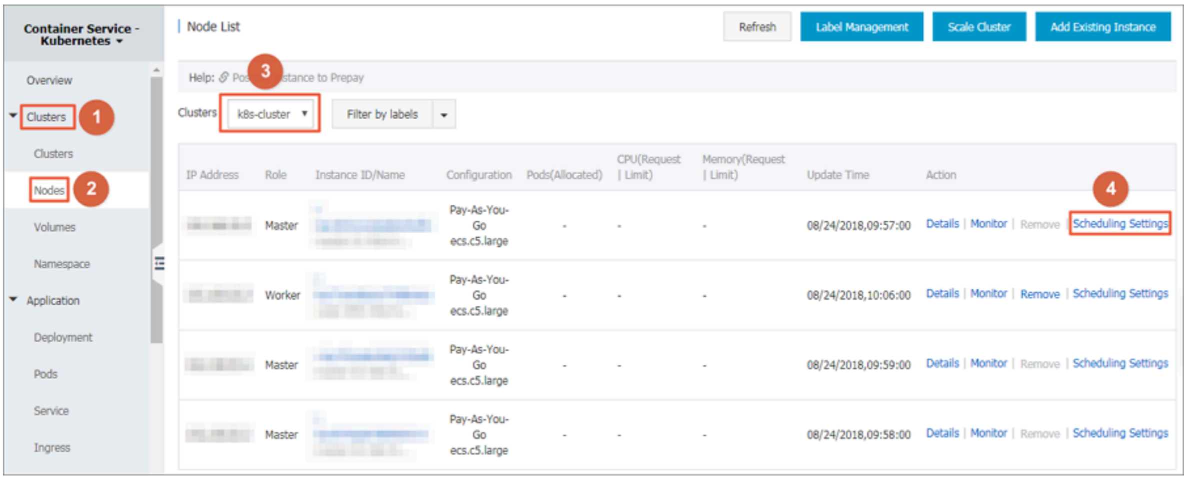
   Click Confirm in the displayed dialog box. The node label is deleted.

1.5.5 Set node scheduling

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

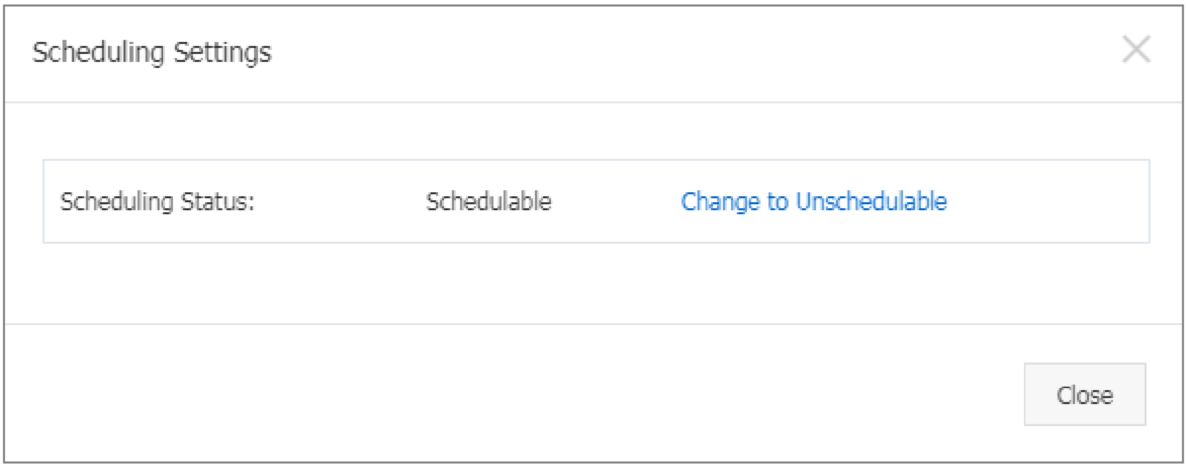
Procedure

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

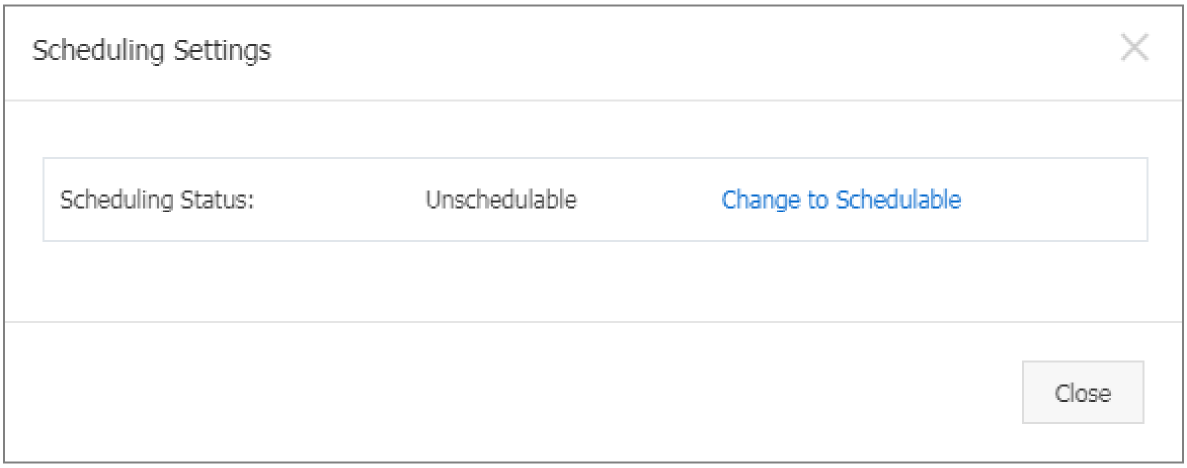


4. Set node scheduling in the displayed dialog box. In this example, click Change to Unschedulable to set the node to unschedulable.

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



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



What's next

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

1.5.6 Remove a node

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

Prerequisites

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

Context

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

Procedure

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

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

   ```
   kubectl drain node-name
   ```

   > Note:
   > The `node-name` parameter must be in the format of `your-region-name.node-id`.
   >
   > - `your-region-name` indicates the name of the region where your cluster resides.
   > - `node-id` indicates the ID of the ECS instance in which the node to be removed resides. For example, `cn-hanghzou.i-xxx`.

2. Set the node to be removed as the non-schedulable node.

   Method 1: Use a command

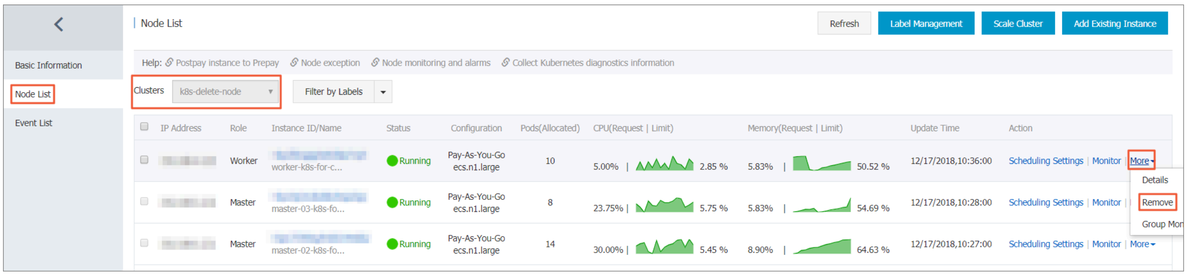
   - Run the following command to set the node to be removed as the non-schedulable node:

     ```
     kubectl cordon node-name
     ```

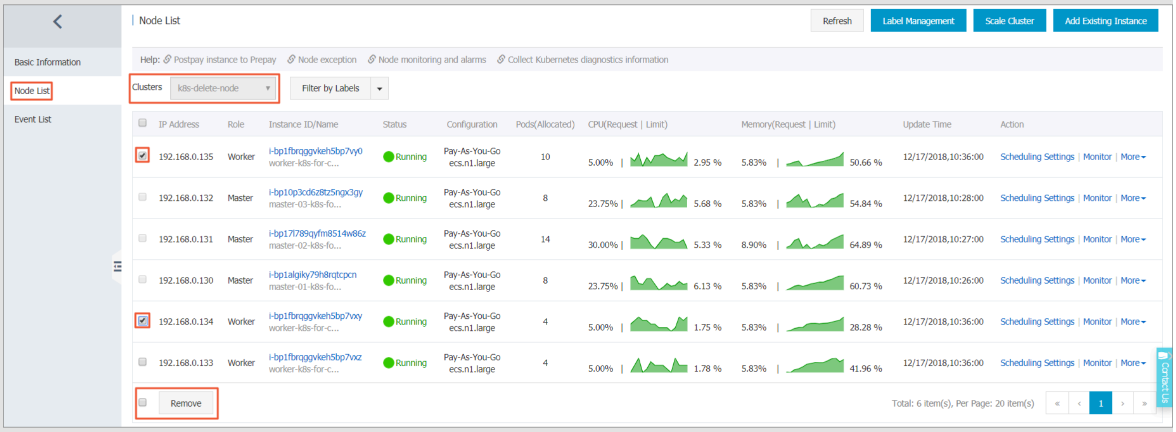
   Method 2: Use the Container Service console

   For more information, see *Set node scheduling*.

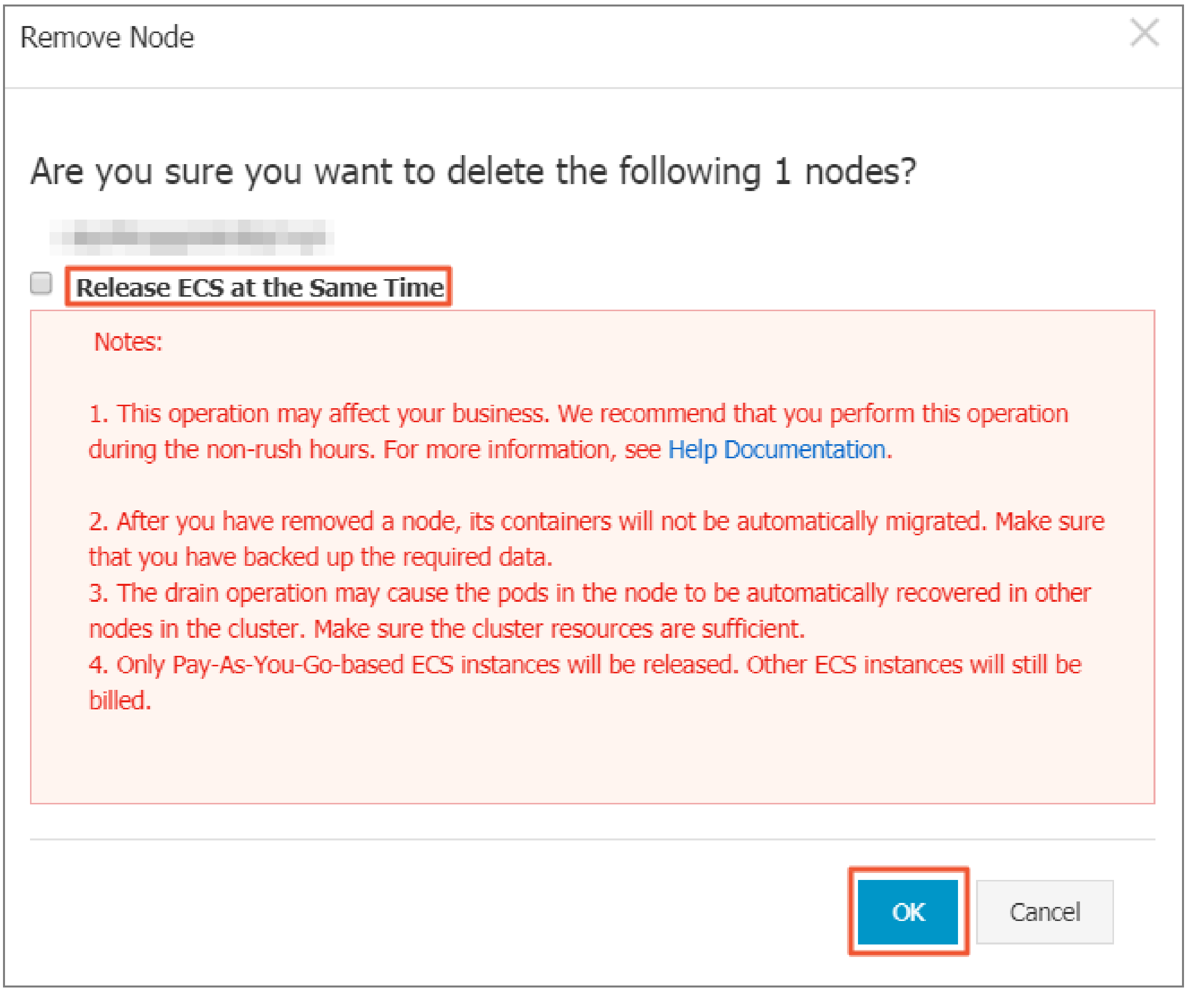
3. In the left-side navigation pane under Kubernetes, choose Clusters > Nodes.
4. Under the target cluster, select the target node, and choose More > Remove in the Action column.



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



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



Note:

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

6. Click OK.

1.5.7 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

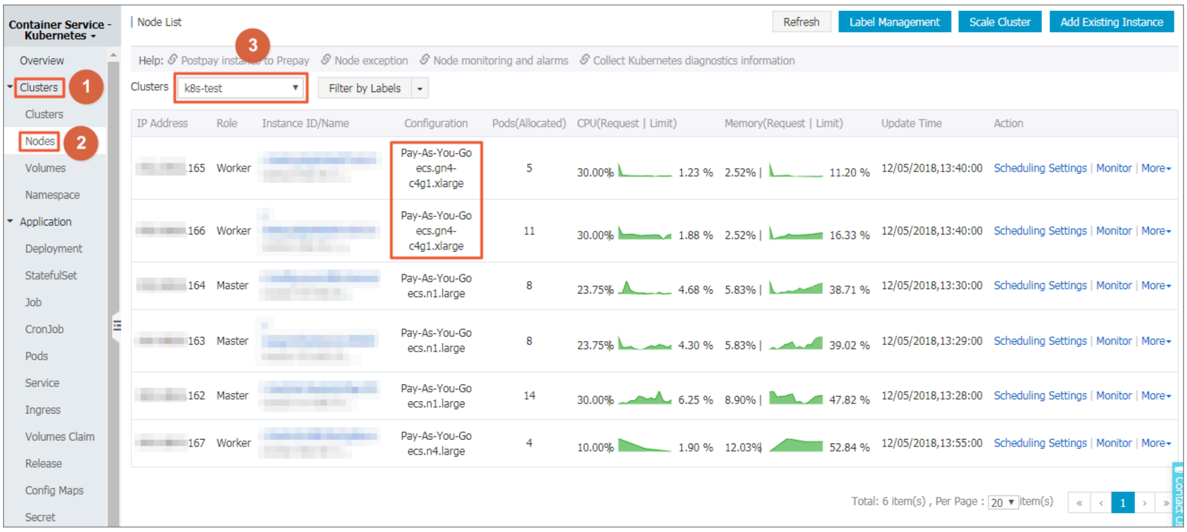
When deploying NVIDIA GPU nodes, Kubernetes that runs on Alibaba Cloud discovers the GPU attribute and exposes it as the node label information. Node labels provide the following benefits:

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

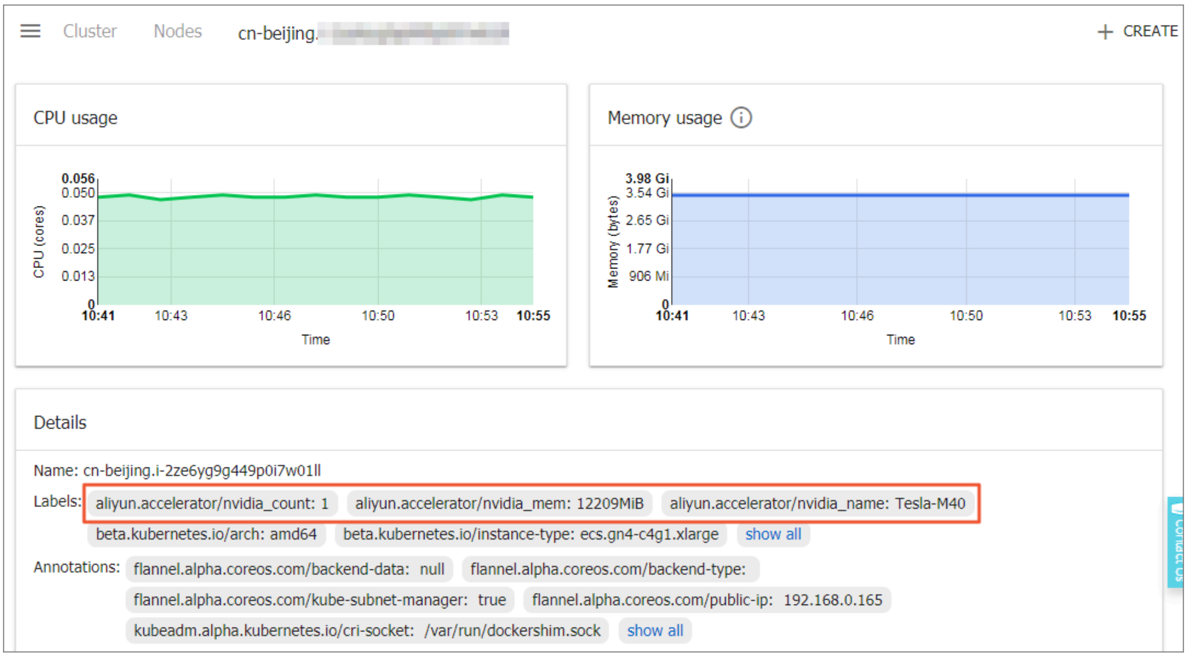
Procedure

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.



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



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

```
# kubectl get nodes
NAME                                 STATUS    ROLES     AGE
VERSION
cn-beijing.i-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:

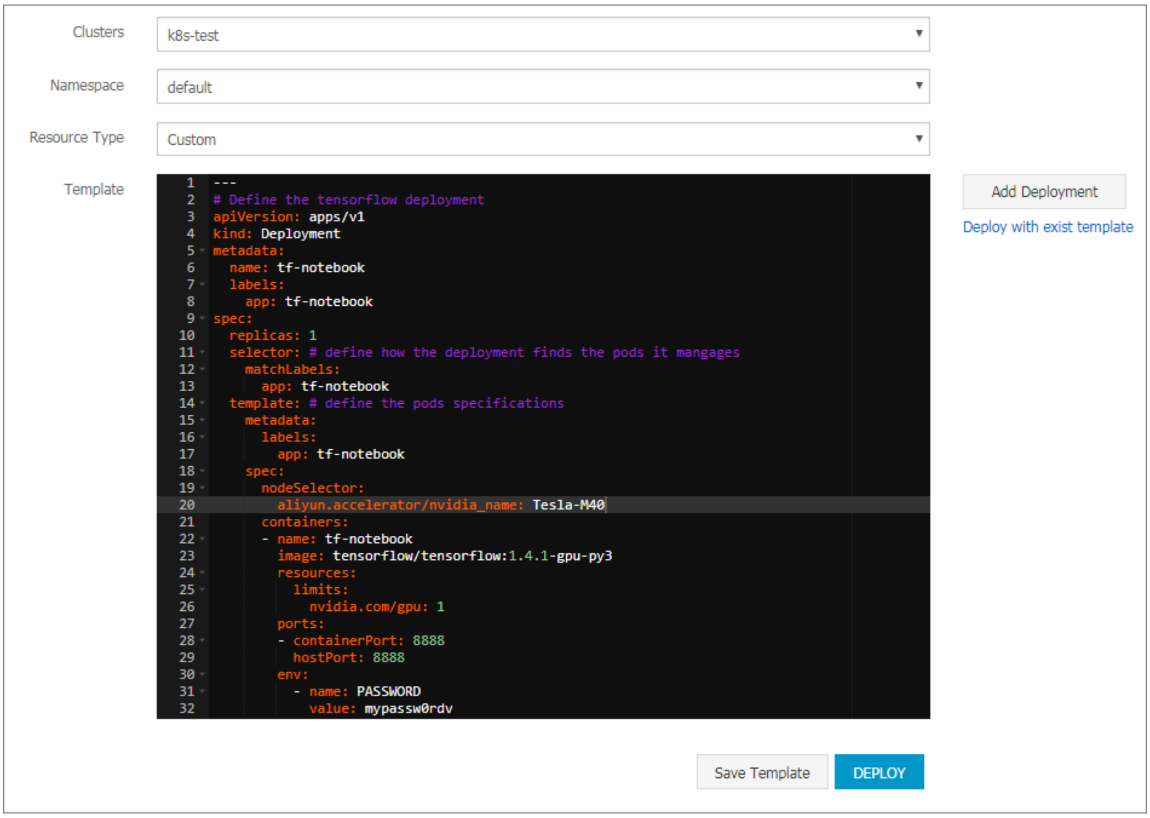
Key	Value
`aliyun.accelerator/nvidia_count`	Number of GPU cores
`aliyun.accelerator/nvidia_mem`	GPU memory in MiB
`aliyun.accelerator/nvidia_name`	Name of the GPU computing card of the NVIDIA device

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.



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:
```

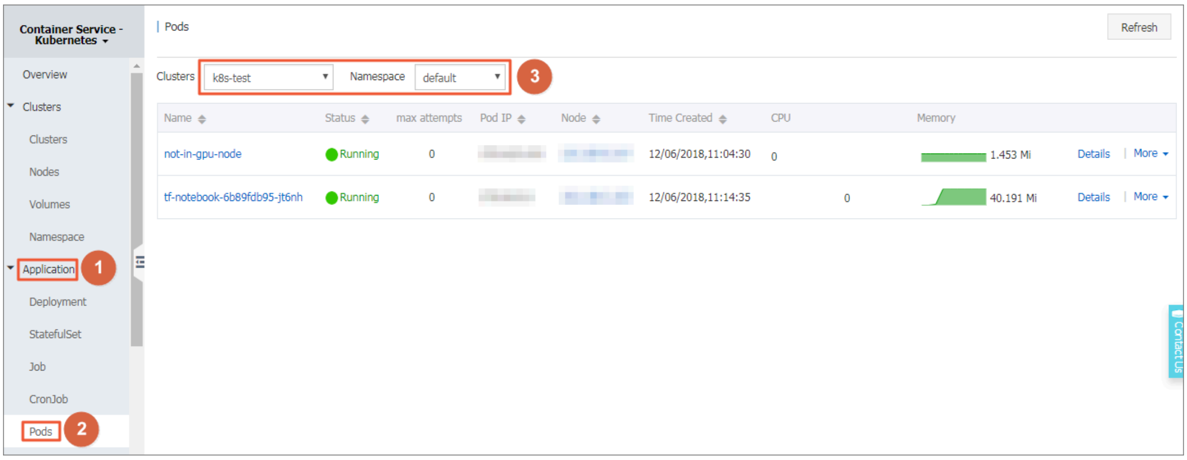
```
          - name: PASSWORD
            value: mypassw0rdv
```

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.



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.

1.5.8 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

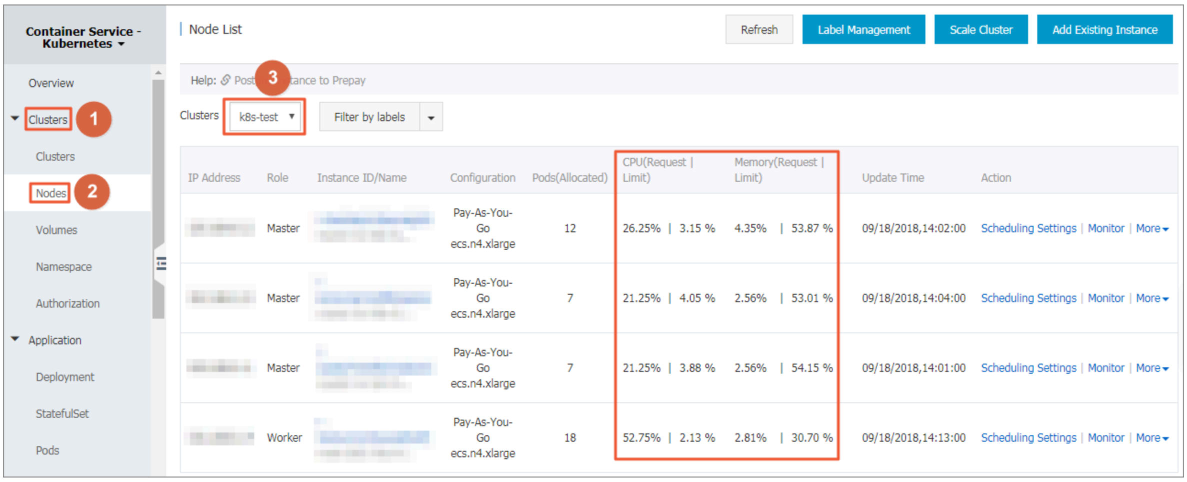
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Clusters > Nodes.

   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.



1.5.9 Mount a disk to a Kubernetes cluster node

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

Prerequisites

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

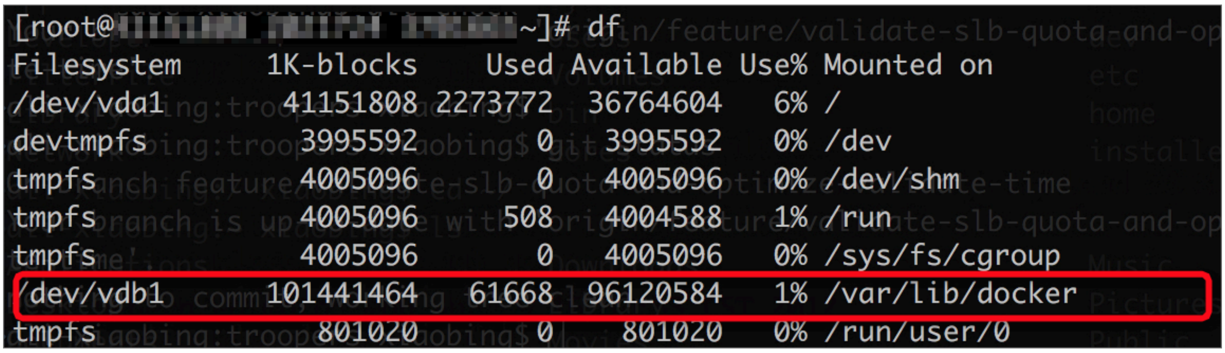
You can mount a disk to an existing Kubernetes cluster node by using either of the following methods:

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

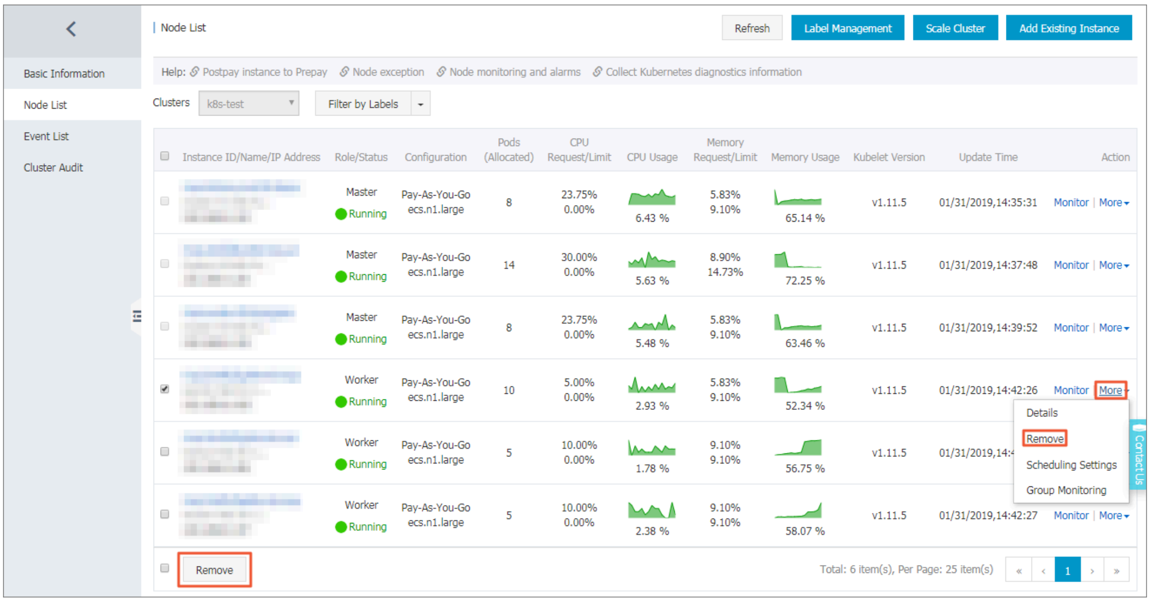
Note:

- We recommend that you create a snapshot of the target node or back up node data to avoid data loss.
- Additionally, you must ensure that you can schedule your cluster applications to other nodes.
- We recommend that you perform this operation during off-peak service hours to avoid disruptions to your business.
- Draining a node reschedules pods on the node to other nodes. Therefore, you must ensure that your Kubernetes cluster has sufficient nodes. We recommend that you add cluster nodes in advance as needed.

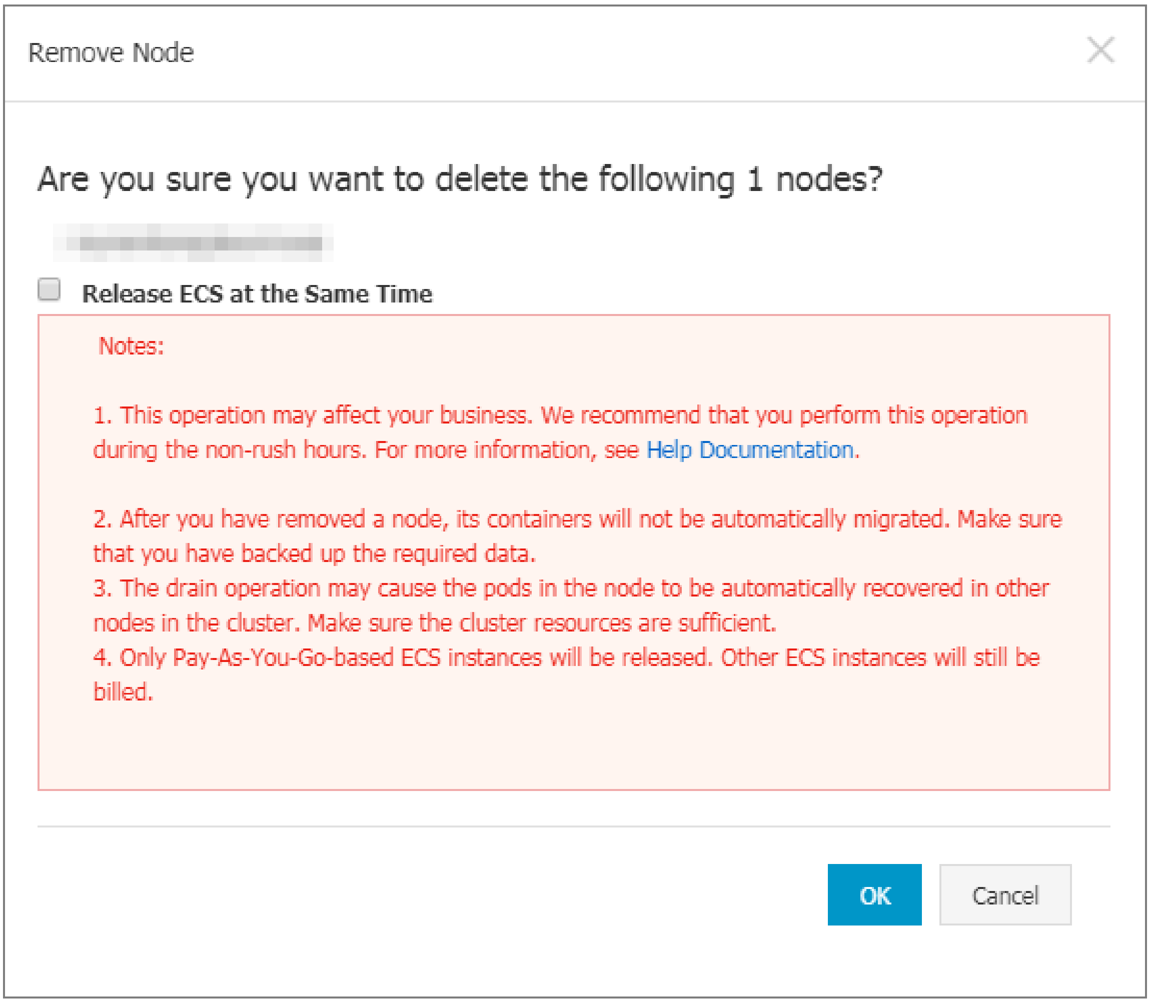
Before performing the operation, you need to determine whether a disk is already mounted to the target cluster node. To do so, run the `df` command on the target Worker node, and then check whether `/var/lib/docker` has been mounted to `/dev/vdb1`. If the disk mounting operation failed, you can mount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



1. Set the target node as unschedulable. For more information, see *Mark node as unschedulable*.
2. Drain the target node. For more information, see *Safely drain a node*.
3. Remove the target node. This topic uses the Container Service console as an example.
   a. Log on to the *Container Service console*.
   b. In the left-side navigation pane, click Node.
   c. Select the target node, and click Remove or choose More > Remove.



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

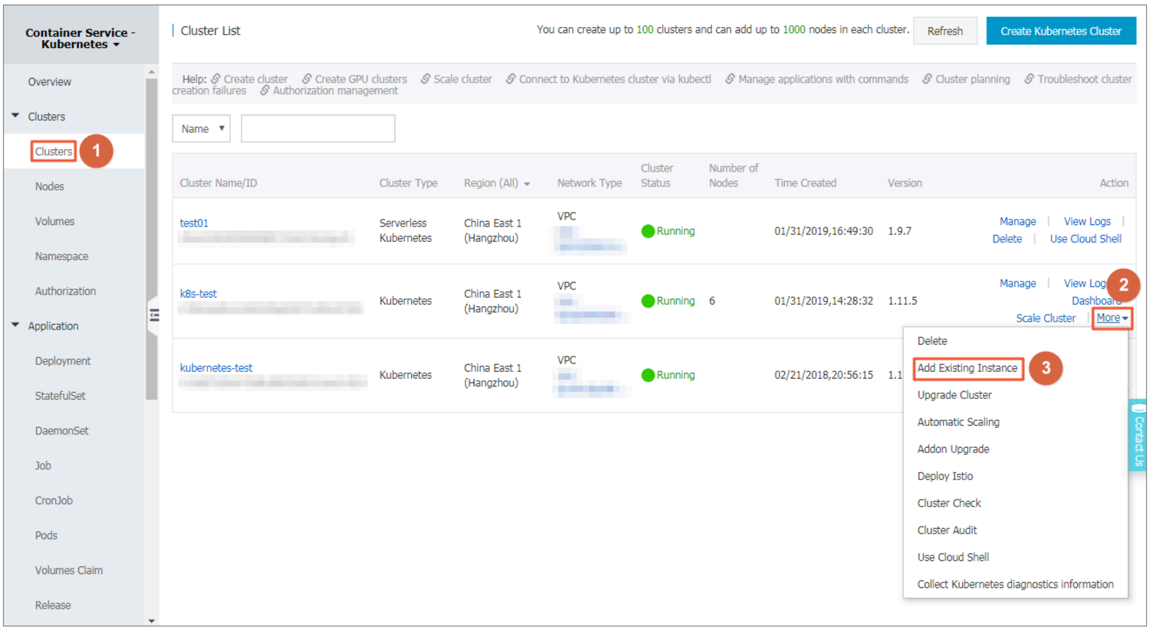


> **Note:**
> We recommend that you do not select the Release ECS at the same time check box. Otherwise, the ECS instance used by the target node will be released.

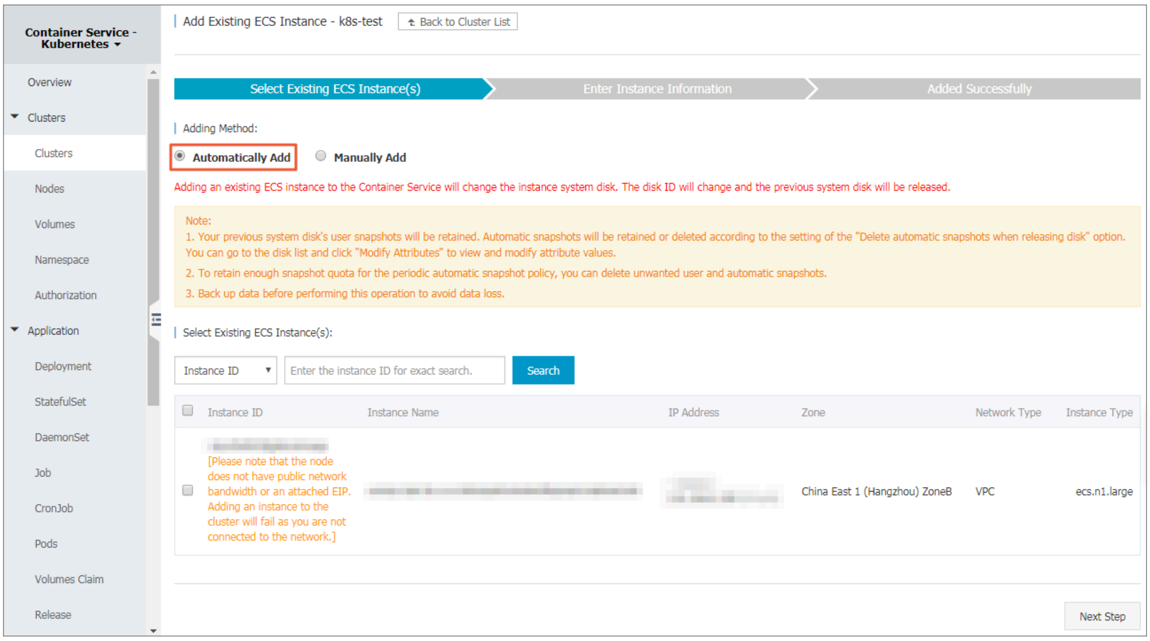
4. Add the removed node to the cluster.

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

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

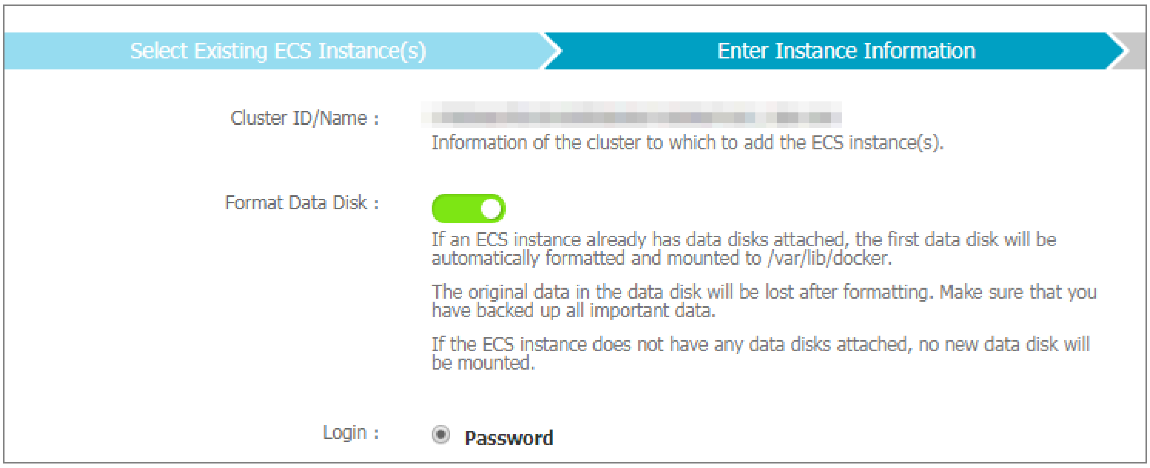


c. Select Automatically Add or Manually Add. In this example, the instance is added automatically.



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

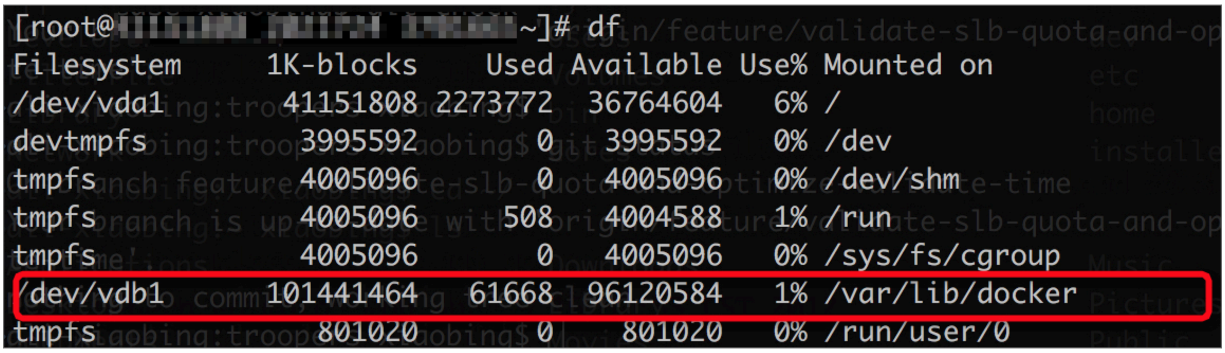
e. Turn on the Format Data Disk switch.



f. Complete other required settings.

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

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



1.5.10 Mount a disk to the Docker data directory

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

Docker data directory

Docker data is stored in disks through a union file system (UnionFS). The default container data and image data of Docker is stored in the *`/var/lib/docker`* directory. You can run the `du` command to view the disk space size occupied by this directory.

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

```
7.9G    /var/lib/docker
```

Scenarios

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

Mount a disk

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

1. Create a disk and mount it to the target ECS instance for which you want to expand the disk capacity.
    a. Log on to the *ECS console* to create a disk.
    b. In the left-side navigation pane, click Instances.
    c. Click the target ECS instance ID.
    d. In the left-side navigation pane, click Disks.
    e. In the upper-right corner, click Mount.
    f. In the displayed dialog box, select the created disk from the target disk drop-down list, and then click OK.
    g. Click Mount to mount the new disk to the target ECS instance, and record the new disk mounting point which is in the format of `/dev/xvd*` or `/dev/vd*`.
2. Log on to the target ECS instance to format the new disk.
    a. Run the `ls -l /dev/xvd*` or `ls -l /dev/vd*` command to verify whether a disk that has the recorded mounting point has been mounted to the ECS instance.
    b. Run the `fdisk` command to partition the new disk, and then run the `mkfs.ext4` command to format the new disk.

```
root@c836831d69e4040e797eff4d3c4dcd983-node2:~# ll /dev/xvd*
brw-rw---- 1 root disk 202,  0 May 26 15:44 /dev/xvda
brw-rw---- 1 root disk 202,  1 May 26 15:44 /dev/xvda1
brw-rw---- 1 root disk 202, 16 May 27 13:03 /dev/xvdb
root@c836831d69e4040e797eff4d3c4dcd983-node2:~# fdisk -S 56 /dev/xvdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x446953ae.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-62914559, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-62914559, default 62914559):
Using default value 62914559

Command (m for help): wq
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@c836831d69e4040e797eff4d3c4dcd983-node2:~# ll /dev/xvd*
brw-rw---- 1 root disk 202,  0 May 26 15:44 /dev/xvda
brw-rw---- 1 root disk 202,  1 May 26 15:44 /dev/xvda1
brw-rw---- 1 root disk 202, 16 May 27 13:08 /dev/xvdb
brw-rw---- 1 root disk 202, 17 May 27 13:08 /dev/xvdb1
root@c836831d69e4040e797eff4d3c4dcd983-node2:~# mkfs.ext4 /dev/xvdb1
mke2fs 1.42.9 (4-Feb-2014)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
1966080 inodes, 7864064 blocks
393203 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
240 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

3. Migrate the Docker data to the new disk.

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

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

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


mkdir /var/lib/docker

mount -a
```

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

4. Start the Docker daemon and kubelet, and check the data location.

a. Run the `service docker start` command to start the Docker daemon, and run the `service kubelet start` command to start kubelet.
b. Run the `df` command to verify whether */var/lib/docker* has been mounted to the new disk. If you need to start the Kubernetes cluster, skip this step.

```
root@c836831d69e4040e797eff4d3c4dcd983-node2:/var/lib# df
Filesystem     1K-blocks    Used Available Use% Mounted on
udev              497280       4    497276   1% /dev
tmpfs             101628     712    100916   1% /run
/dev/xvda1      41151808 1928420  37109960   5% /
none                   4       0         4   0% /sys/fs/cgroup
none                5120       0      5120   0% /run/lock
none              508136     288    507848   1% /run/shm
none              102400       0    102400   0% /run/user
/dev/xvdb1      30831612  667168  28575248   3% /var/lib/docker
```

c. Run the `docker ps` command to check whether containers are lost. Restart containers as needed. For example, you can restart a container that has not been set the `restart:always` label.

```
root@c836831d69e4040e797eff4d3c4dcd983-node2:/var/lib# docker ps
CONTAINER ID        IMAGE                                                 COMMAND                  CREATED          STATUS
4f564091bffa        registry.aliyuncs.com/acs/logspout:0.1-41e0e21        "/bin/logspout"          21 hours ago     Up 3 minutes
gspout_2
a5aba5fbedae        registry.aliyuncs.com/acs/ilogtail:0.9.9              "/bin/sh -c 'sh /usr/"   21 hours ago     Up 3 minutes
gtail_2
5e3d8fe154bb        registry.aliyuncs.com/acs/monitoring-agent:0.7-1cf85e6   "acs-mon-run.sh --hel"   21 hours ago     Up 3 minutes
_acs-monitoring-agent_1
fb72c2388b0e        registry.aliyuncs.com/acs/volume-driver:0.7-252cb09   "acs-agent volume_exe"   21 hours ago     Up 3 minutes
er_volumedriver_2
604fcb4ad720        registry.aliyuncs.com/acs/routing:0.7-c8c15f0         "/opt/run.sh"            21 hours ago     Up 3 minutes
uting_1
8fe1d6ed15b5        registry.aliyuncs.com/acs/agent:0.7-6967e86           "acs-agent join --nod"   21 hours ago     Up 3 minutes
999da3883264        registry.aliyuncs.com/acs/tunnel-agent:0.21           "/acs/agent -config=c"   21 hours ago     Up 3 minutes
```

5. If a container has been migrated to other nodes, you can schedule it back to the target node to which you mounted the new disk.

For more information, see *Container Service*.

1.6 Namespace management

1.6.1 Create a namespace

This topic describes how to create a namespace.

Prerequisites

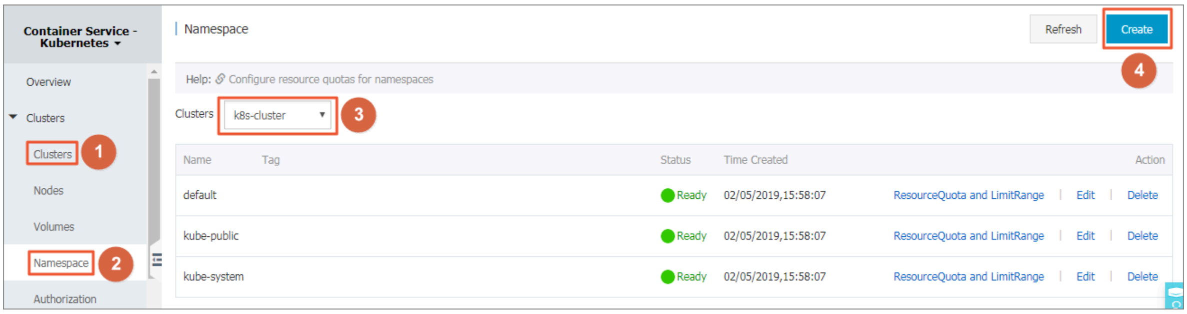
You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

Context

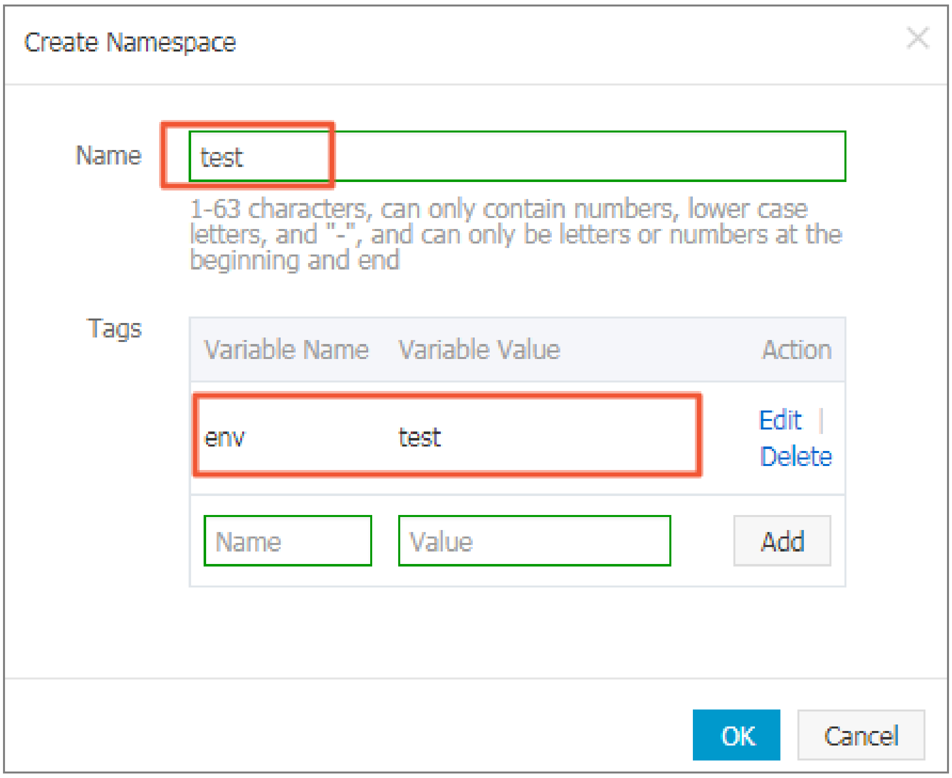
In a Kubernetes cluster, you can use namespaces to create multiple virtual spaces. When a large number of users share a cluster, multiple namespaces can be used to effectively divide different work spaces and assign cluster resources to different tasks. Furthermore, you can use *resource quotas* to assign resources to each namespace.

Procedure

1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Namespace.
3. Select the target cluster, and then click Create in the upper-right corner.



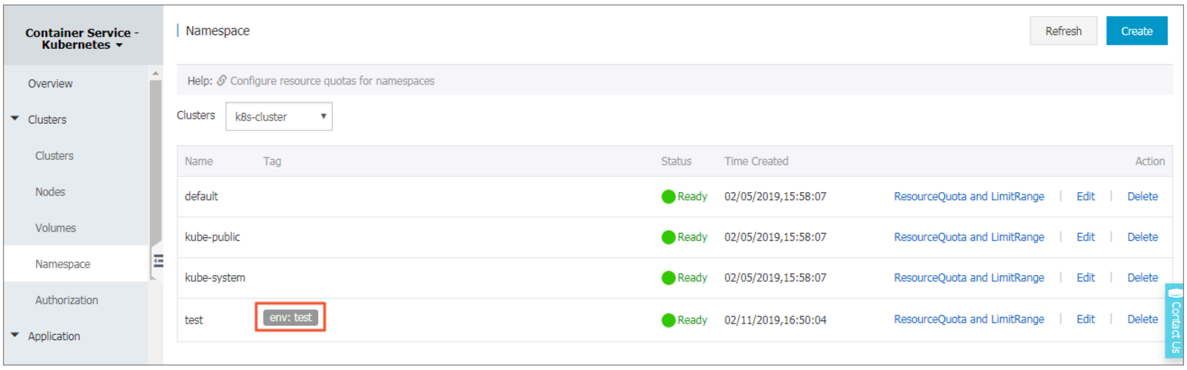
4. In the displayed dialog box, set a namespace.



- Name: Enter a name for the namespace name. The name must be 1 to 63 characters in length and can contain numbers, letters, and hyphens (-). It must start and end with a letter or number. In this example, test is used as the name.
- Tags: Add one or multiple tags to the namespace to identify the characteristics of the namespace. For example, you can set a tag to identify that this namespace is used for the test environment.

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

5. Click OK.
6. The namespace named test is displayed in the namespace list.



1.6.2 Set resource quotas and limits for a namespace

This topic describes how to set resource quotas and limits for a namespace through the Container Services console.

Prerequisites

- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a namespace. In this topic, a namespace named test is used. For more information, see *Create a namespace*.
- You have connected to the Master node of the cluster. For more information, see *Connect to a Kubernetes cluster by using kubectl*.

Context

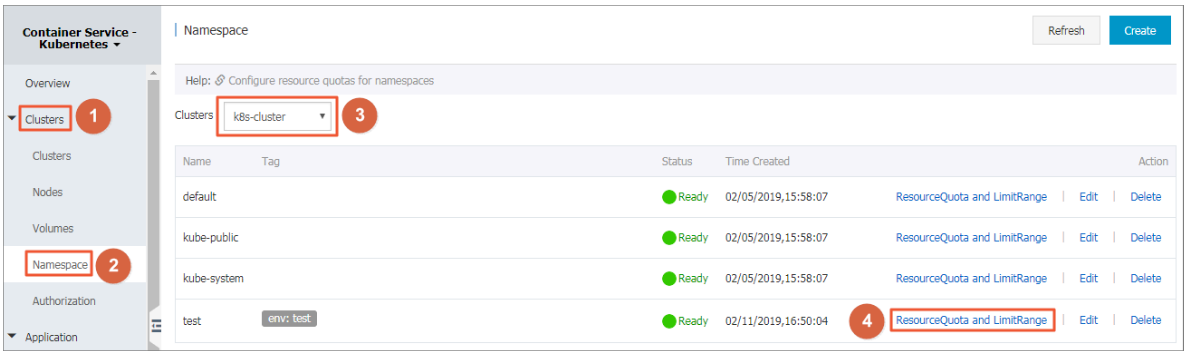
By default, a running pod uses the CPU and memory resources of nodes without limit . That is, any pod can use the computing resources of the cluster without restraint. Therefore, pods of a namespace may deplete the cluster resources.

Namespaces can be used as virtual clusters to serve multiple users. Therefore, setting resource quotas for a namespace is regarded as a best practice.

For a namespace, you can set the quotas of resources, such as CPU, memory, and number of pods. For more information, see *Resource quotas*.

Procedure

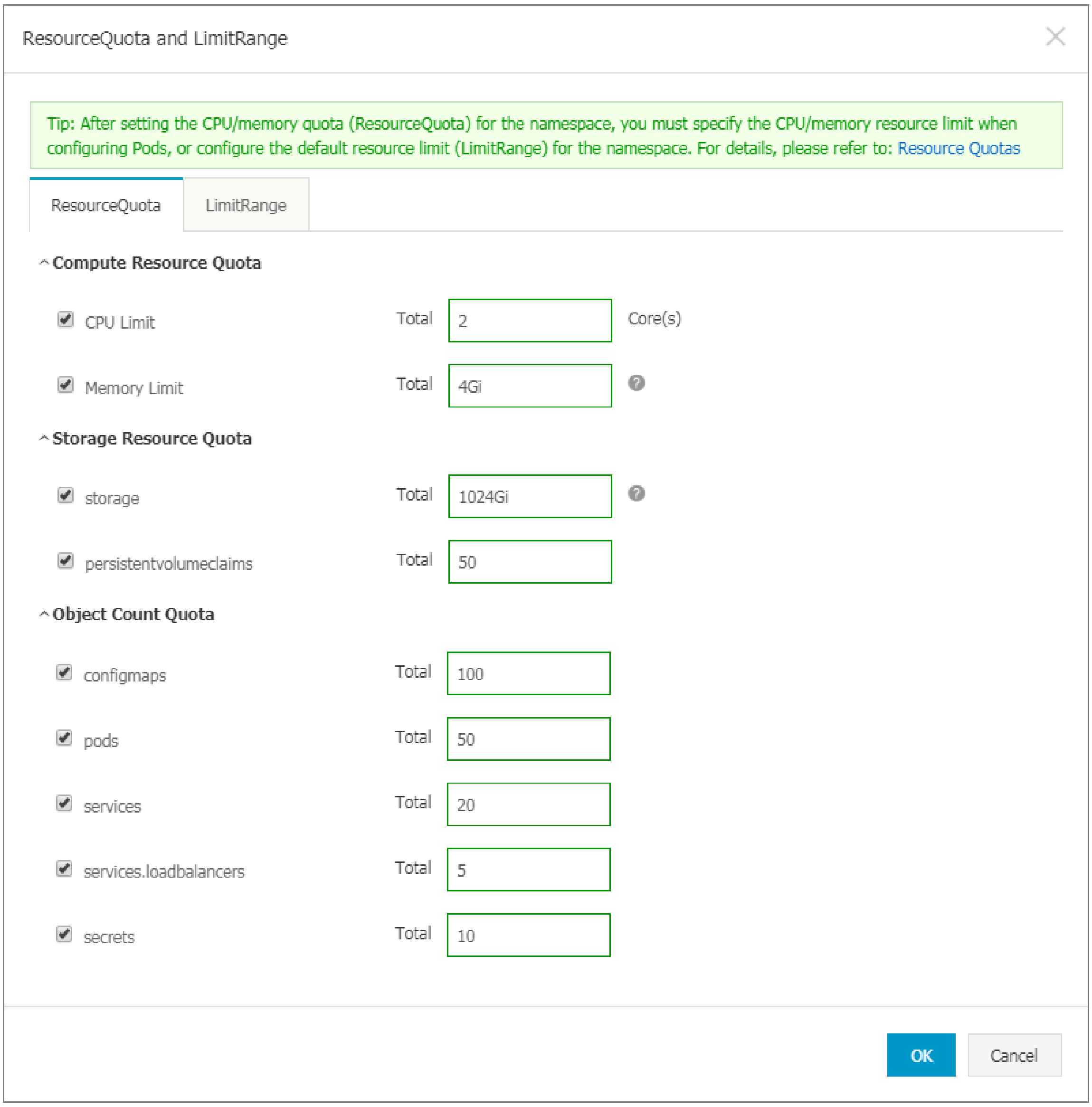
1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Namespace. Select the target cluster and click ResourceQuota and LimitRange on the right of the test namespace.



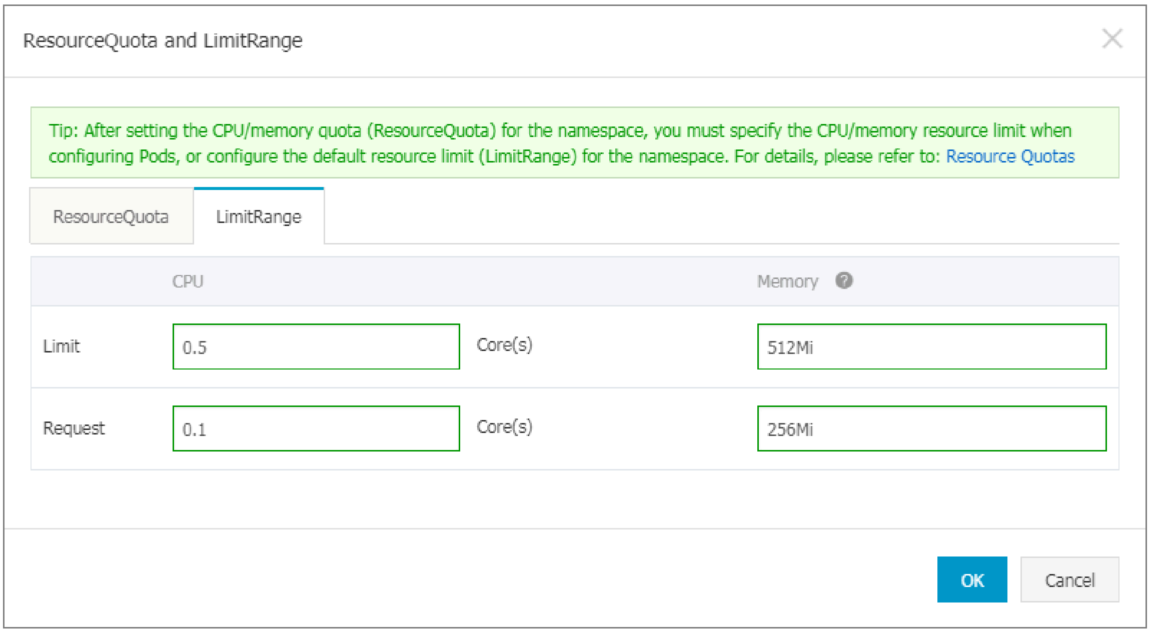
3. In the displayed dialog box, set resource quotas and default resource limits.

> **Note:**
> After setting CPU/memory quotas for a namespace, you must specify CPU/memory resource limits or set the default resource limits for the namespace when creating a pod. For more information, see *Resource quotas*.

a) Set resource quotas for the namespace.



b) To control the amount of resources consumed by containers, set resource limits and resource requests for containers in this namespace. For more information, see *https://kubernetes.io//memory-default-namespace/*.



4. Connect to the Master node and then run the following commands to view the resources of the test namespace:

```
#kubectl get limitrange,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
```

1.6.3 Edit a namespace

This topic describes how to edit a namespace.

Prerequisites

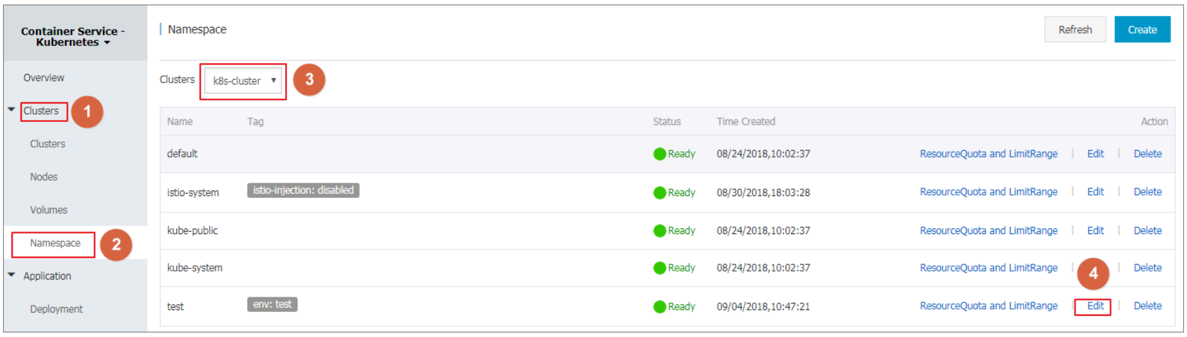
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a namespace. In this topic, a namespace named test is used. For more information, see *Create a namespace*.

Context

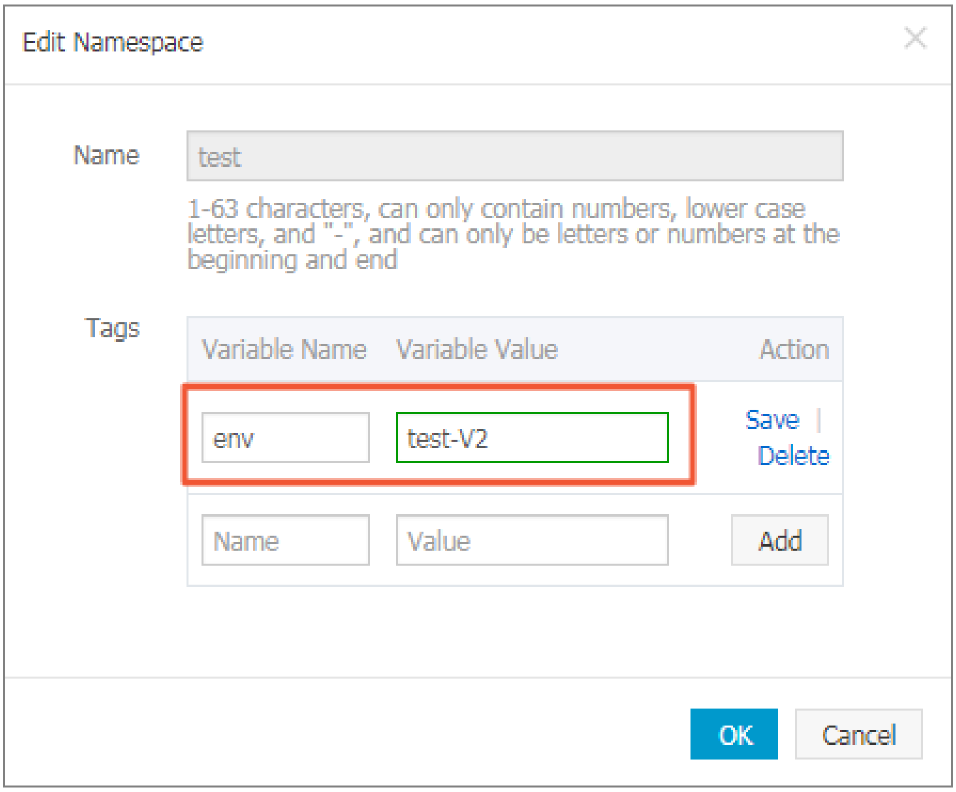
Editing a namespace means to add, modify, or delete the details of a namespace tag.

Procedure

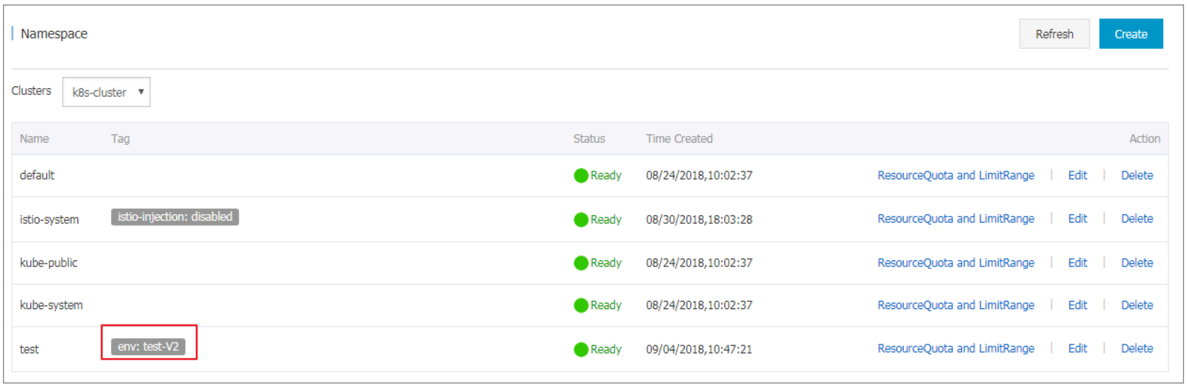
1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Namespace.
3. Select the target cluster and then click Edit on the right of the target namespace tag.



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



5. Click OK. The edited namespace tag is then displayed in the namespace list.



1.6.4 Delete a namespace

This topic describes how to delete a namespace you no longer require.

Prerequisites

- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a namespace. In this topic, a namespace named test is used. For more information, see *Create a namespace*.

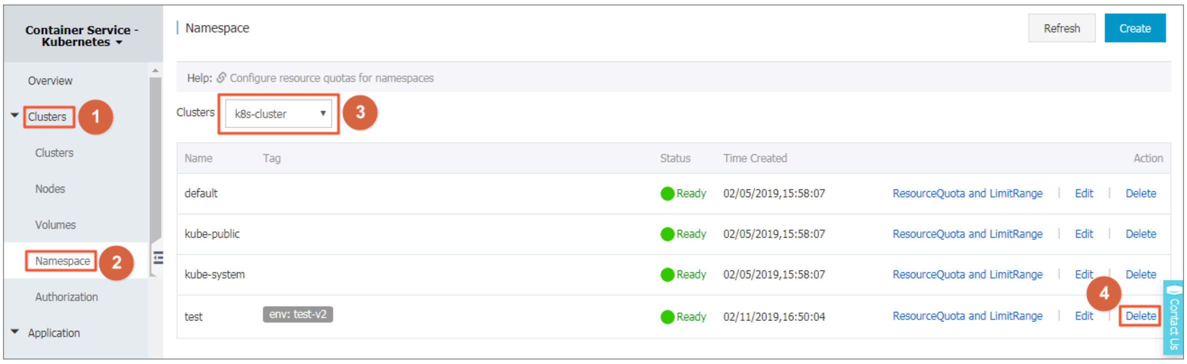
Context

Note:

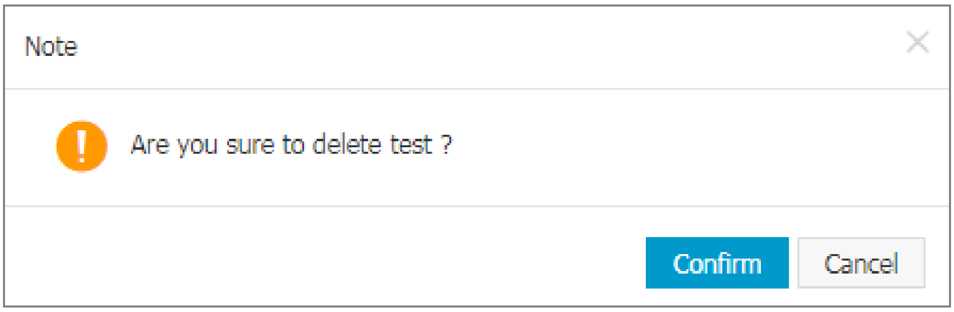
Deleting a namespace also deletes all of its resource objects. Exercise caution when performing this action.

Procedure

1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Namespace.
3. Select the target cluster and then click Delete on the right of the cluster.



4. In the displayed dialog box, click Confirm.



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

1.7 Application management

1.7.1 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

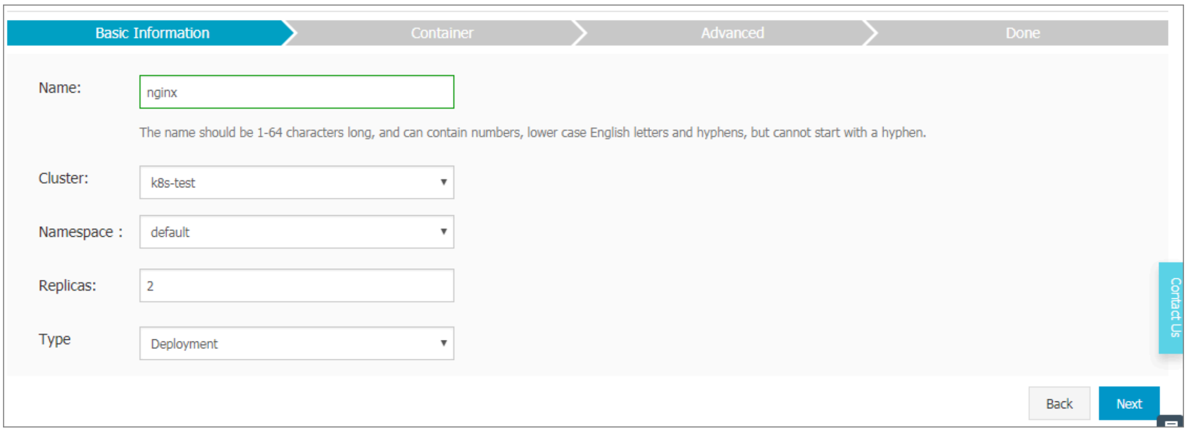
Create a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

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.

4. Configure containers.

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

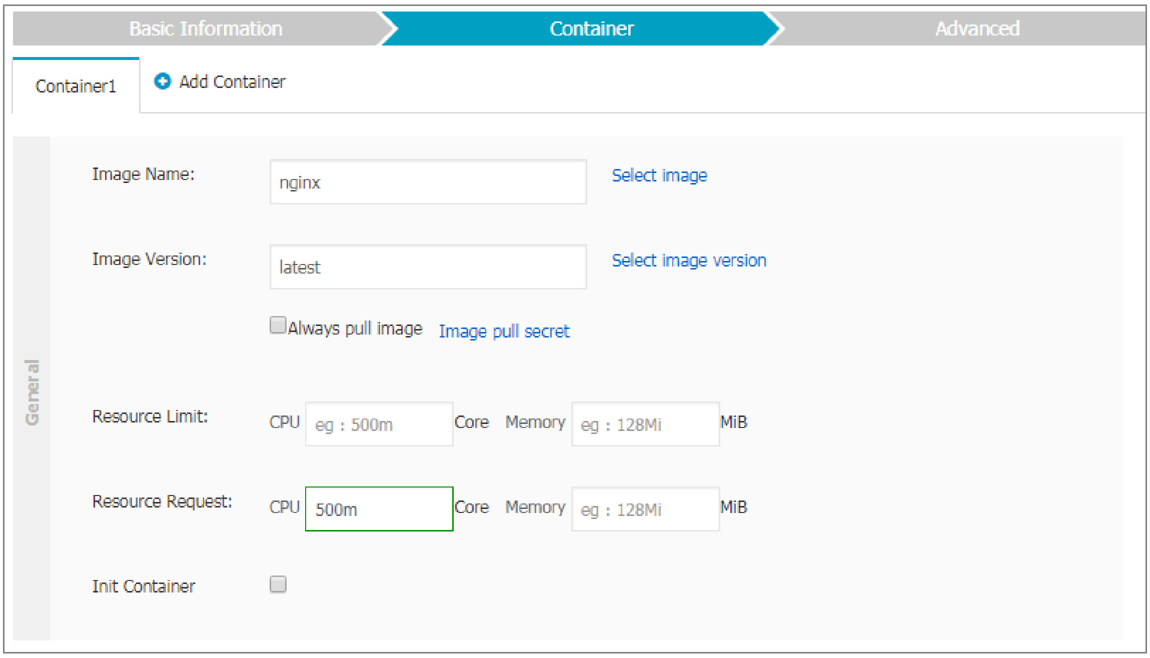
   a) Configure the general settings for the application.

      - 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/*.

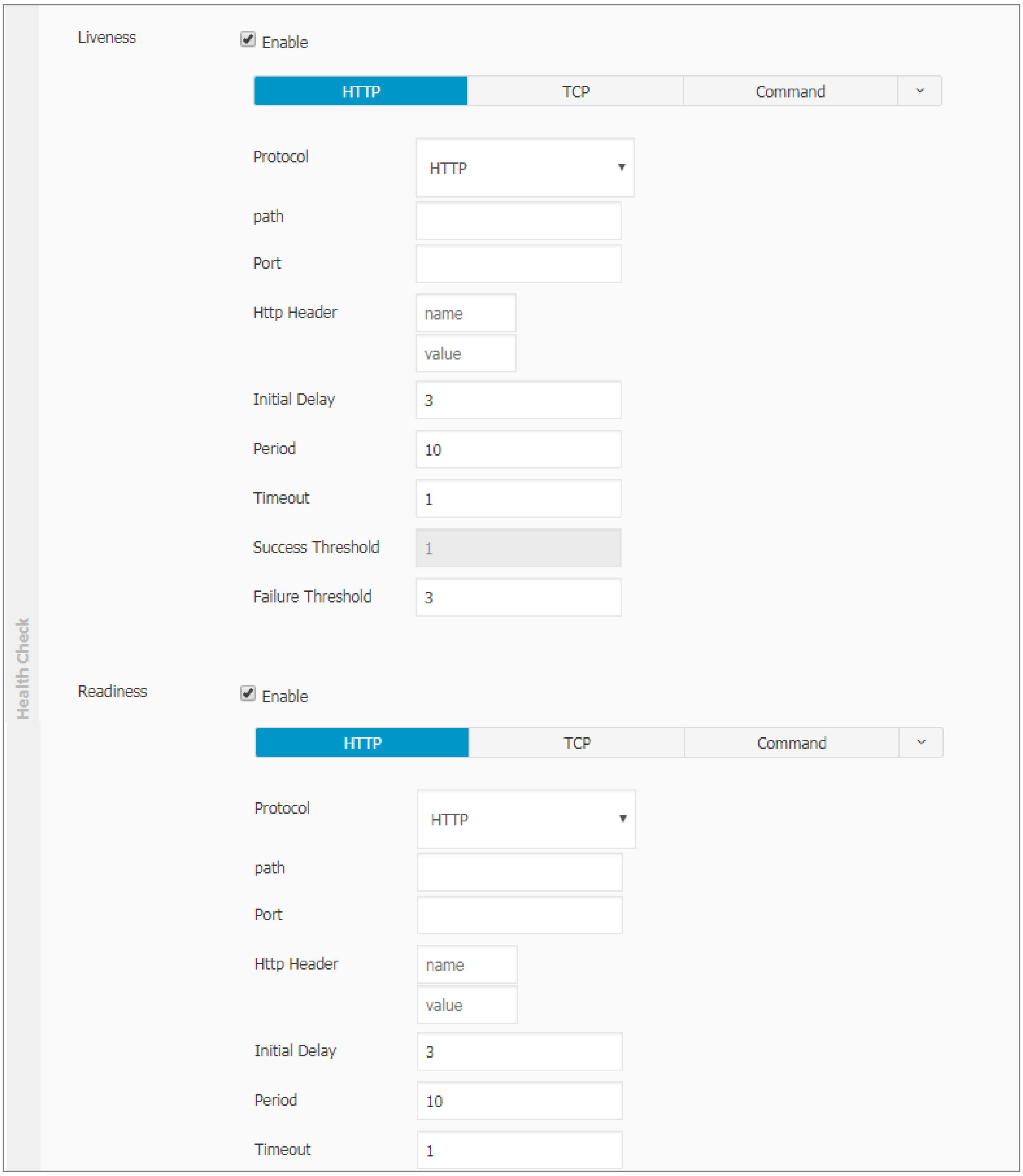


b) Optional: Configure environment variables.

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*.



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 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 configuration. · Initial Delay (in seconds): Namely, the initialDelaySeconds. Seconds for the first probe has to wait after the container is started. The default is 3. · Period (in seconds): Namely, the periodseconds. Intervals at which the probe is performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): Namely, the timeoutSeconds. The time of probe timeout. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes that are considered as successful after a failed probe. The default is 1 and the minimum is 1. It must be 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes that are considered as failed after a successful probe. The default value is 3. The minimum value is 1.

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

Request method	Configuration description
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. · 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.

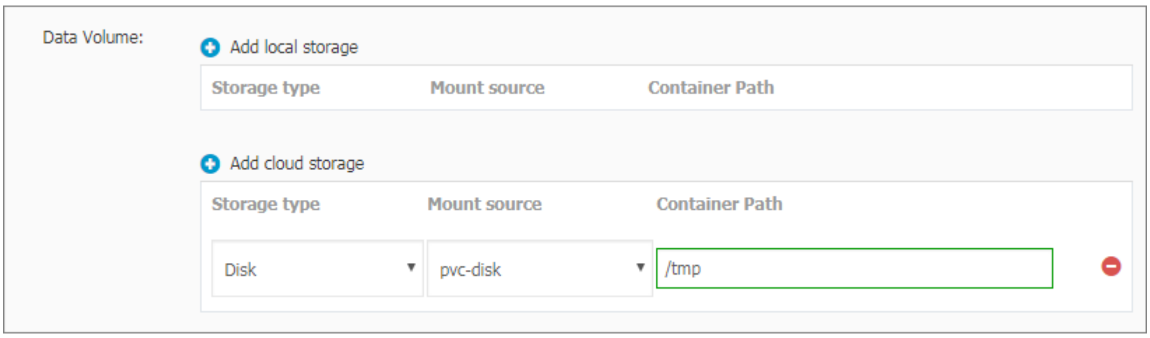
- Start: Configure a pre-start command and parameter for the container.
- Post Start: Configure a post-start command for the container.
- Pre Stop: Configure a pre-end command for the container.

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.



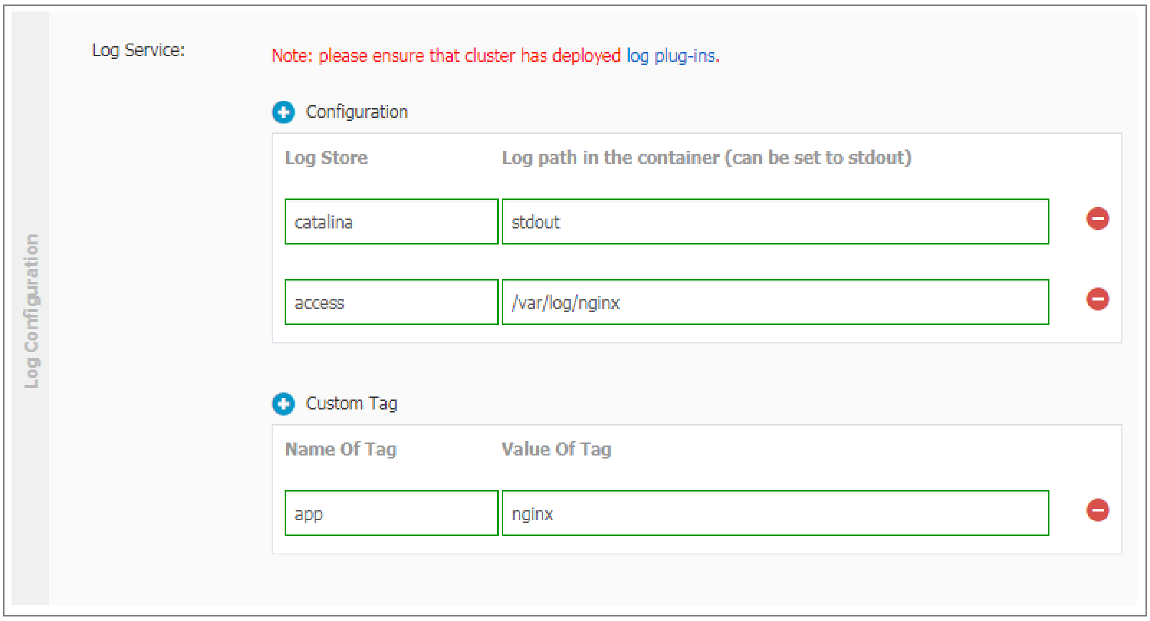
f) Optional: Configure Log Service. You can configure collection methods and customize tags for this service.

> **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.



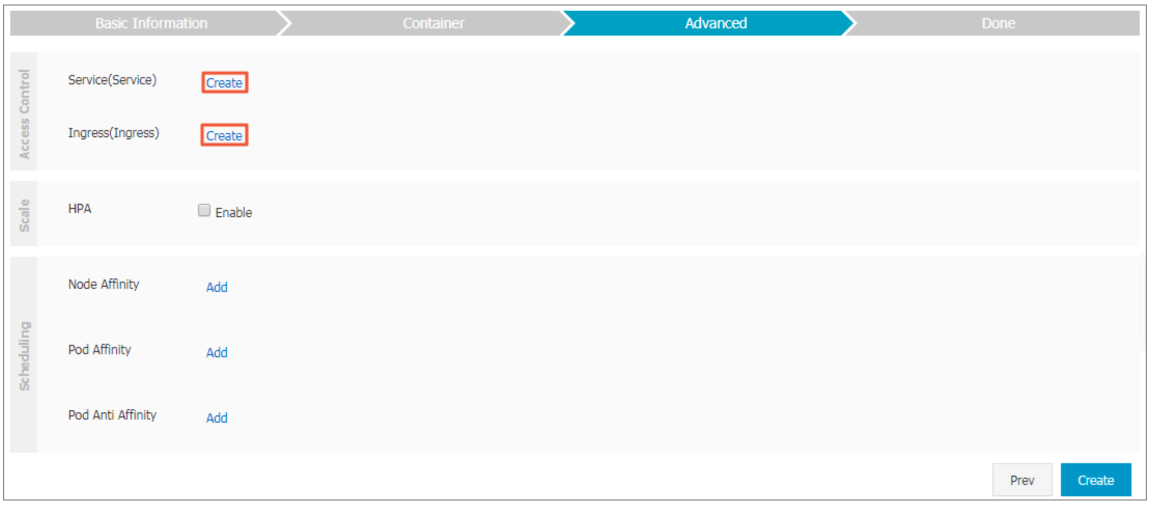
5. Click Next after completing the configurations.
6. Configure advanced settings.
   a) Configure Access Control.

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.

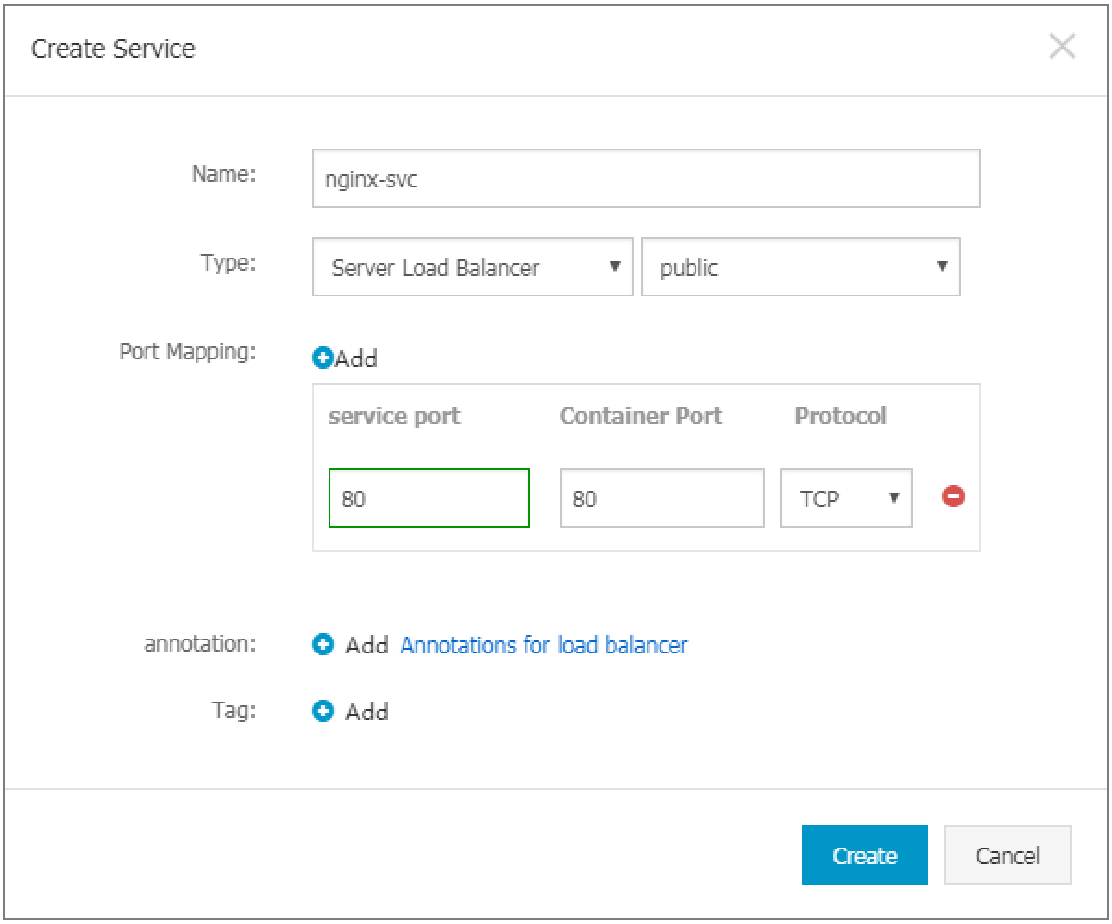
Note:

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/*.



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



- 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.

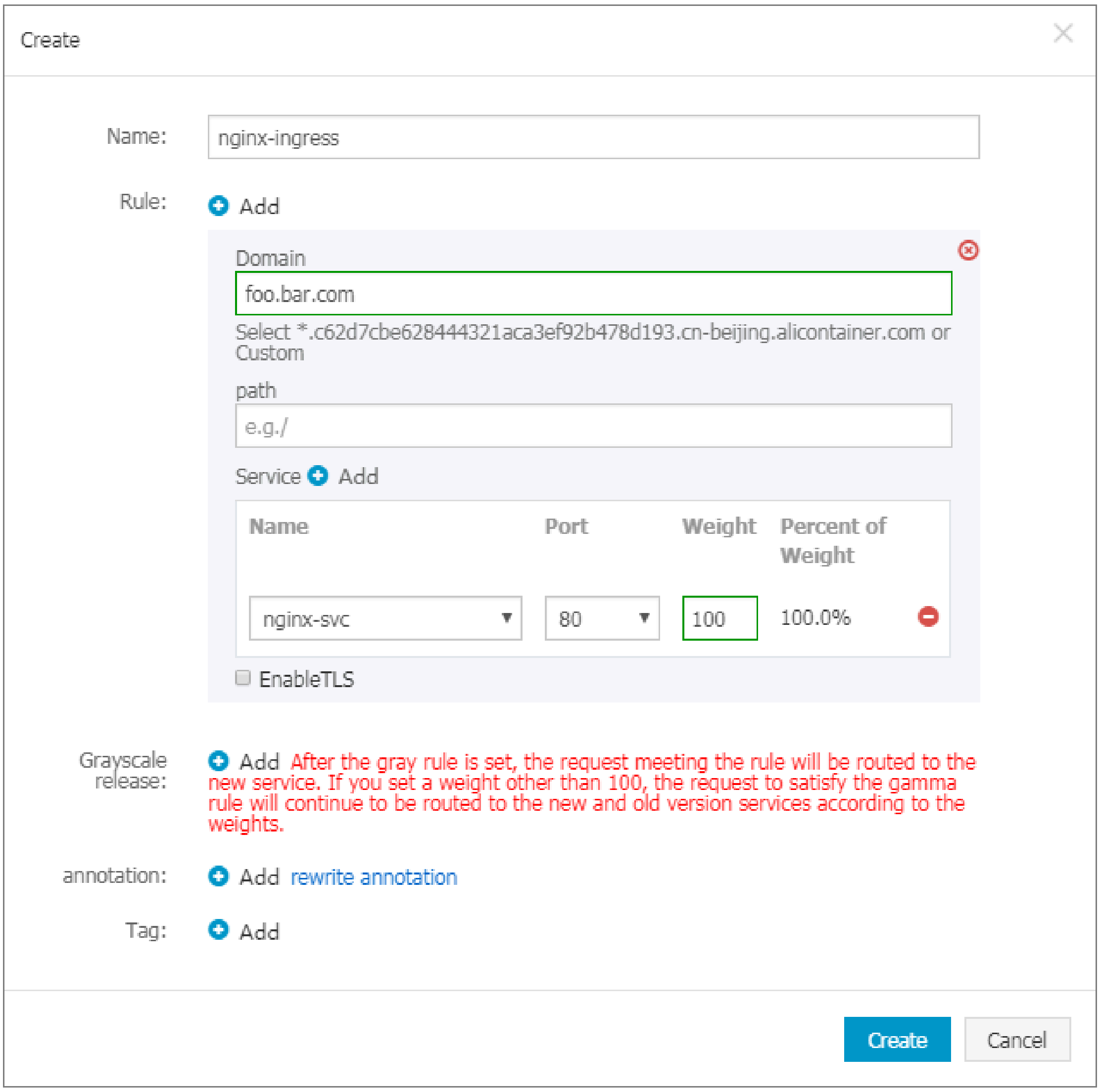
B. Click Create at the right of Ingress. Configure rout rules for the backend pod in the displayed dialog box, and then click Create. For more information about route configuration, see *Ingress configurations*.

Note:

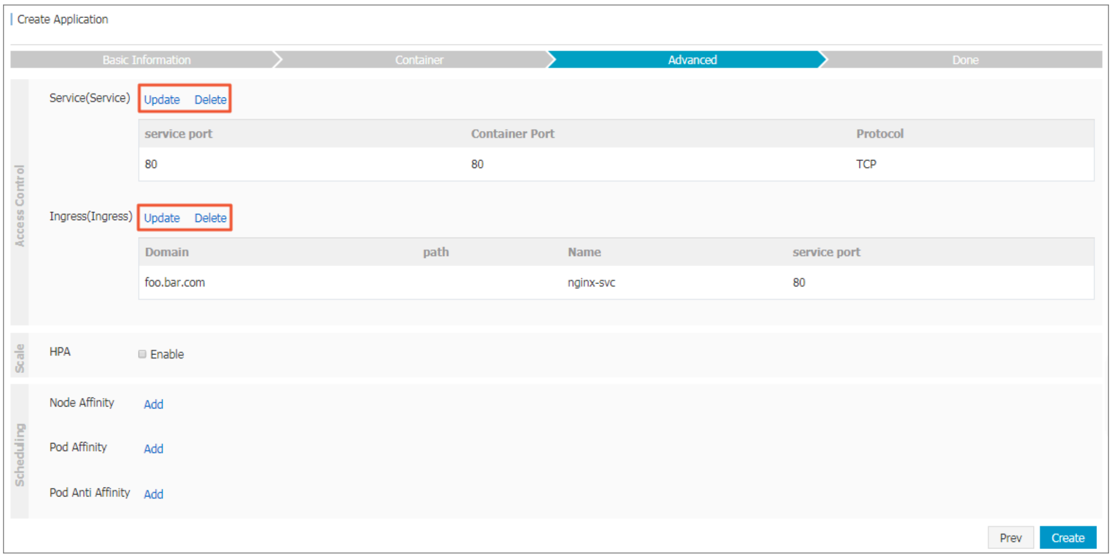
When you create an application by using an image, you can create ingress for only one service. In this example, use a virtual host name as the testing

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
```

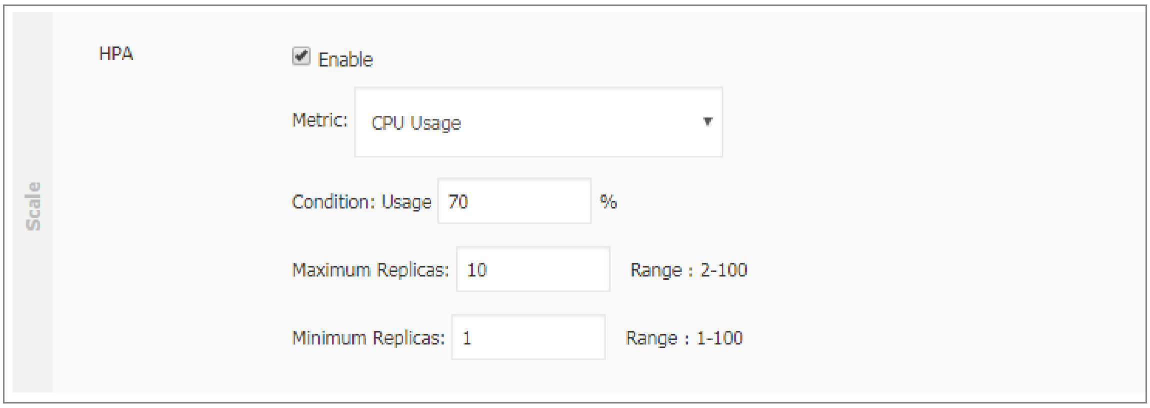


C. The created service and ingress are displayed in the access control section. You can reconfigure the service and ingress by clicking Update and Delete.



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.



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.

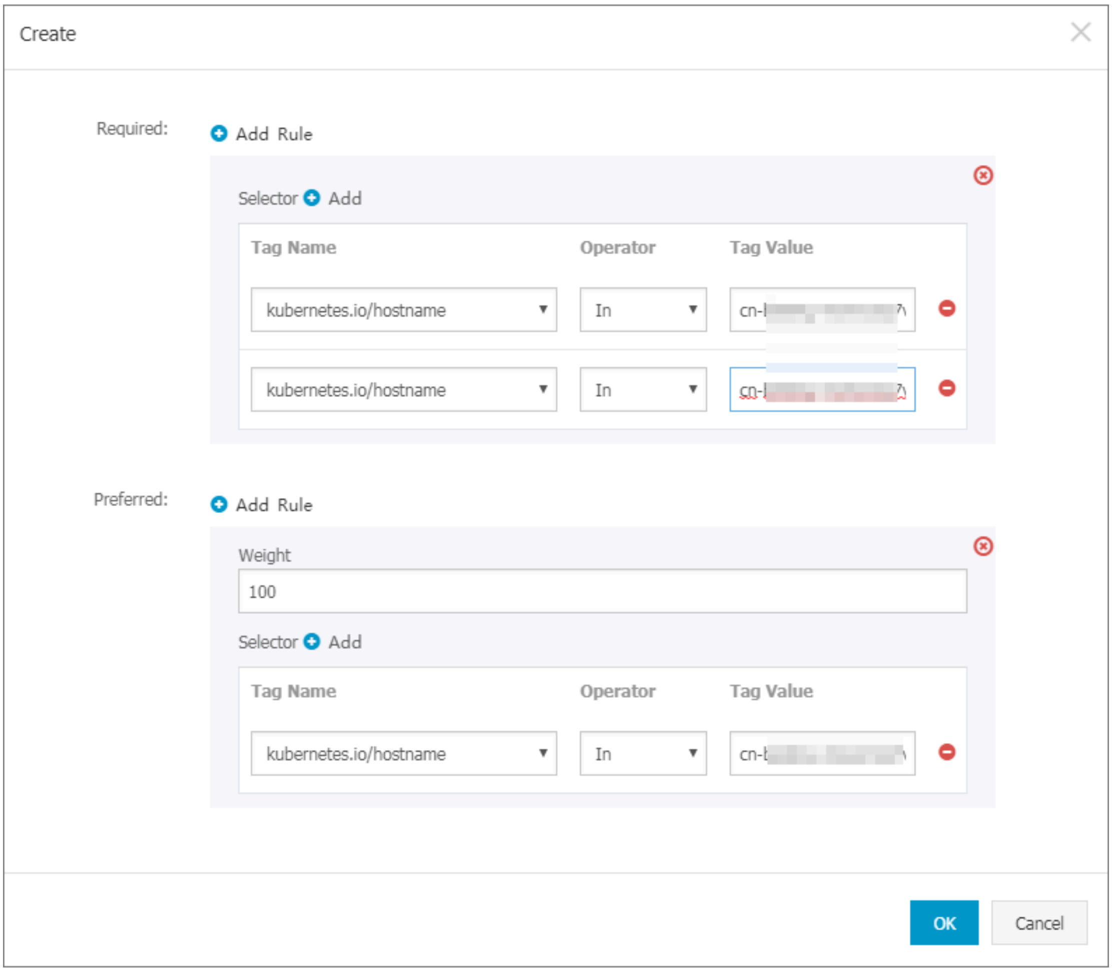
- Minimum Replicas: The minimum number of replicas that the deployment can contract to.

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.

A. Set Node Affinity by configuring node tags.

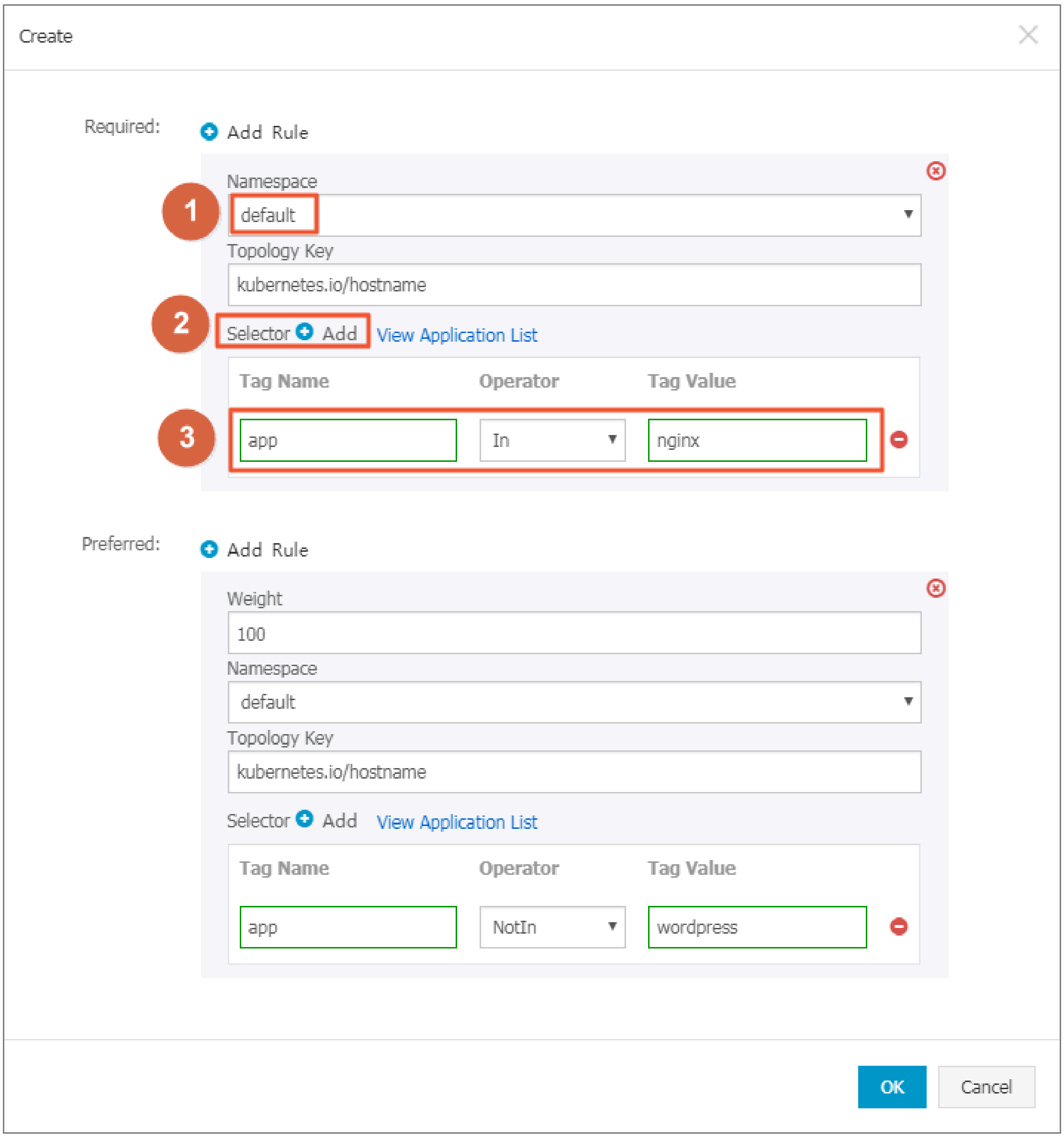


Node scheduling supports both required and preferred rules, and various operators such as In, NotIn, Exists, DoesNotExist, GT, and LT.

- 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.

B. 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.



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.

C. 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.

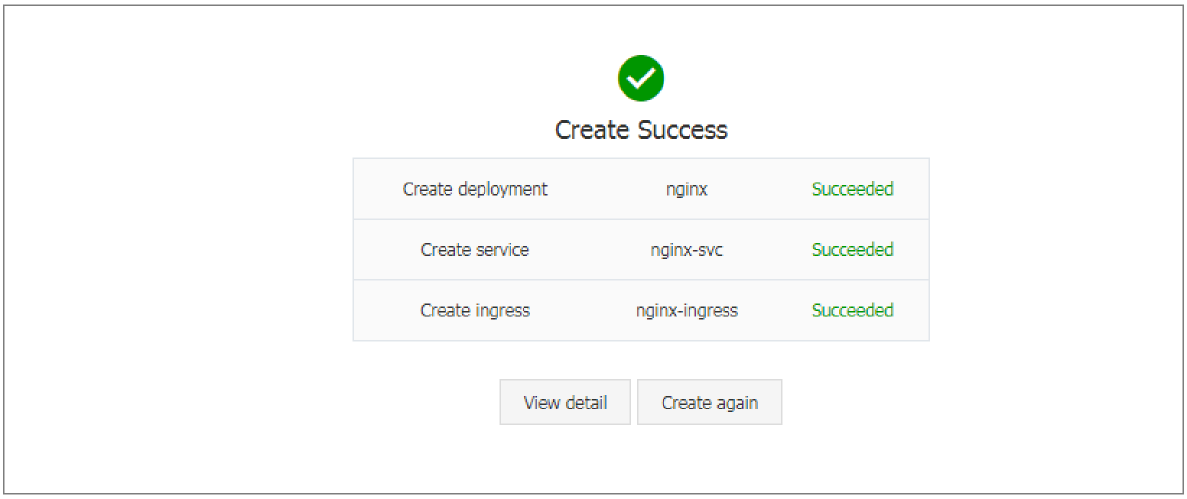
· Grant a pot the exclusive access to a node so as to guarantee no other pods use resources of the node.

· Distribute pods of services that may affect each other to different hosts.

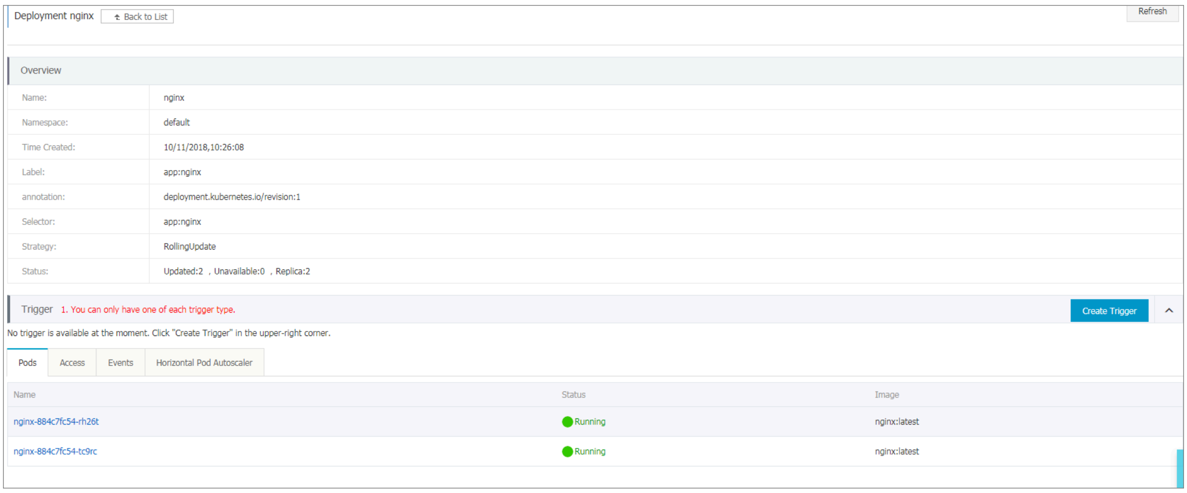
> Note:
> Configuration methods of pod anti affinity scheduling are the same as that of pod affinity. But the same scheduling rules have different meanings for

pod anti affinity scheduling. Select an appropriate scheduling rule based on scenarios.

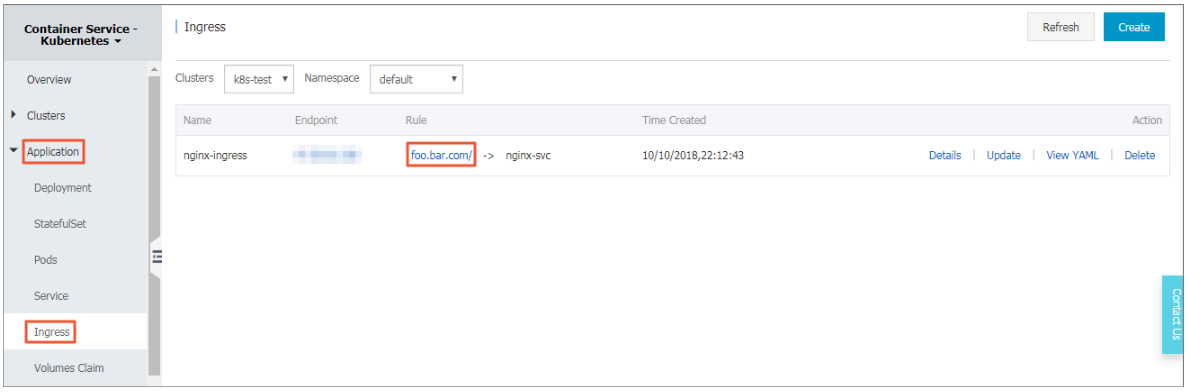
7. Click Create.
8. After you create the application, the create success page is displayed and objects contained in the application are listed by default. You can click View detail to view the deployment details.



The nginx-deployment page is displayed by default.



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



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

1.7.2 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:

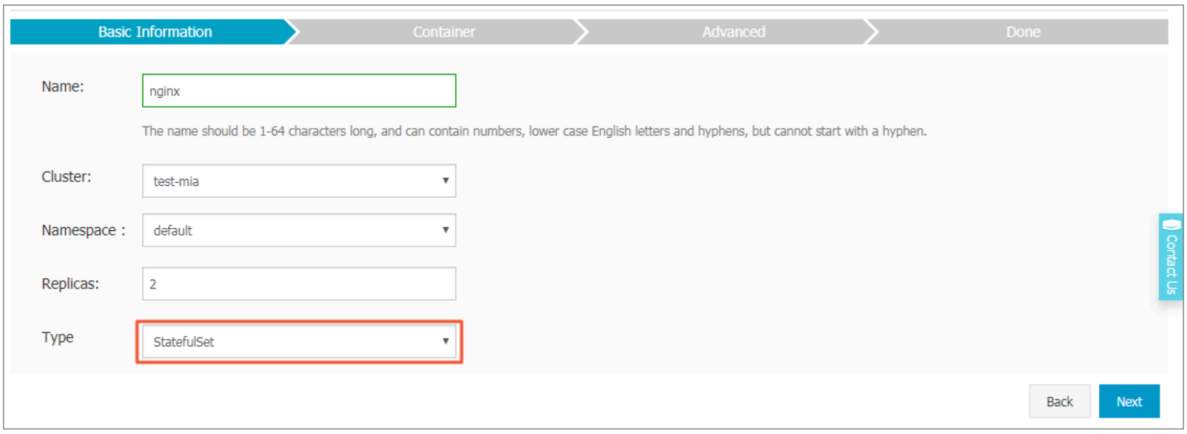
Scenarios	Description
Pod consistency	Contains order (such as startup and stop order) and network consistency. This consistency is related to pods and has nothing to do with the node to which the pods are to be scheduled.
Stable persistent storage	Create a PV for each pod through VolumeClaimTemplate. 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.



4. Configure containers.

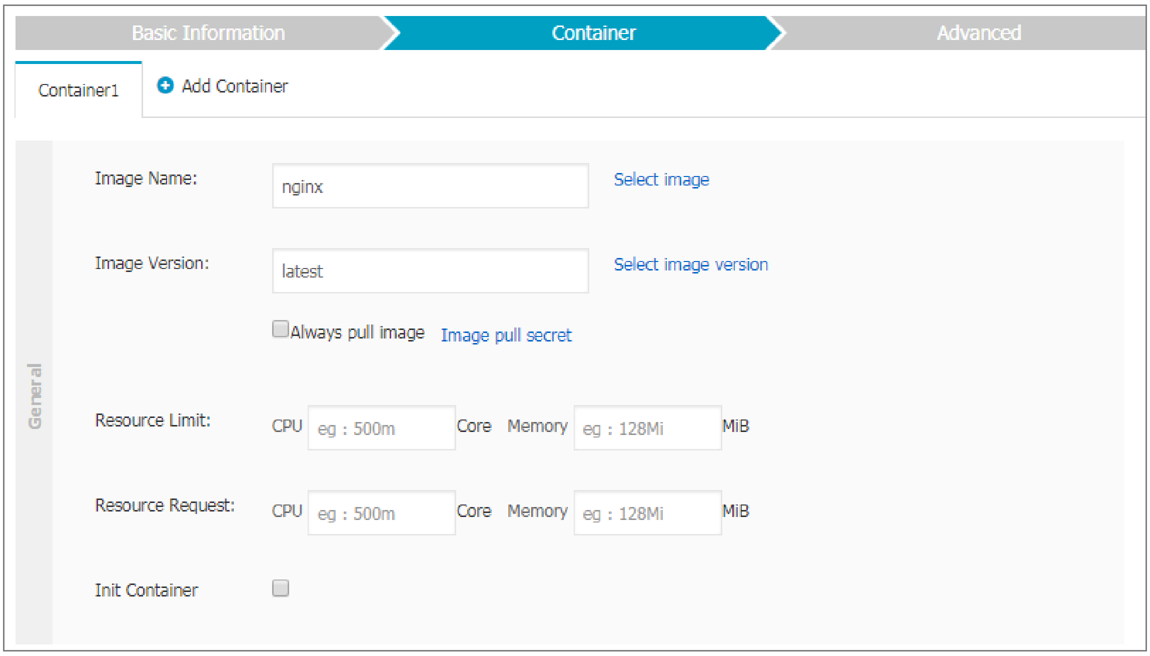
> **Note:**
> You can configure multiple containers for the pod of the application.

a) Configure the general settings for the application.

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

  You can also enter the private registry in the format of `domainname/namespace/imagename:tag` to specify an image.
- Image Version: Click Select image version to select a version. If the image version is not specified, the system uses the latest version by default.
- Always pull image: Container Service caches the image to improve deployment efficiency. During deployment, if the image tag is found consistent with that on the local cache, the image on the local cache is reused and is not pulled again. Therefore, if you do not modify the image tag when changing your codes and image for convenience of upper-layer business, the early image on the local cache is used in the application deployment. With this check box selected, Container Service ignores the cached image and re-pulls the image from the repository when deploying the application to make sure the latest image and codes are always used.
- Resource Limit: Specify the upper limit for the resources (CPU and memory) that can be used by this application to avoid occupying excessive resources. CPU is measured in millicores, that is, one thousandth of one core. Memory is measured in bytes, which can be Gi, Mi, or Ki.

- Resource Request: Specify how many resources (CPU and memory) are reserved for the application, that is, these resources are exclusive to the container. Other services or processes will compete for resources when the resources are insufficient. By specifying the Resource Request, the application will not become unavailable because of insufficient resources.
- Init Container: Selecting this check box creates an Init Container which contains useful tools. For more information, see *https://kubernetes.io/docs/concepts/workloads/pods/init-containers/*.

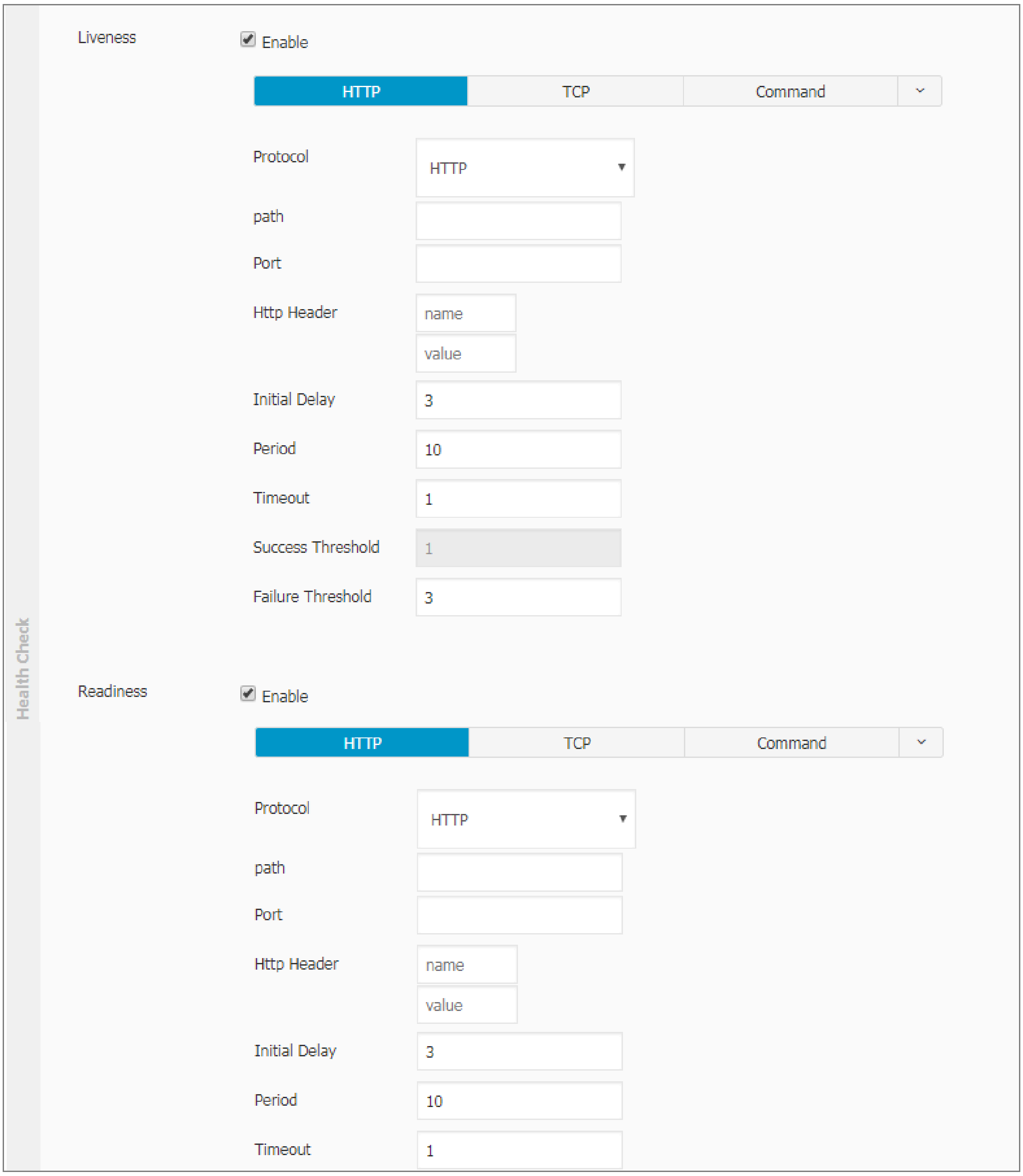


b) Optional: Configure Environment .

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

c) Optional: Configure Health Check.

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


Health Check
Liveness
Enable
HTTP
TCP
Command
Protocol
HTTP
path
Port
Http Header
name
value
Initial Delay
3
Period
10
Timeout
1
Success Threshold
1
Failure Threshold
3
Readiness
Enable
HTTP
TCP
Command
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. · HTTP Header: Custom headers in the HTTP request. HTTP allows repeated headers. Supports the correct configuration of key values. · Initial Delay (in seconds): Namely, the initialDelaySeconds. Seconds for the first probe has to wait after the container is started. The default is 3. · Period (in seconds): Namely, the periodseconds. Intervals at which the probe is performed. The default value is 10. The minimum value is 1. · Timeout (in seconds): Namely, the timeoutSeconds. The time of probe timeout. The default value is 1 and the minimum value is 1. · Success Threshold: The minimum number of consecutive successful probes that are considered as successful after a failed probe. The default is 1 and the minimum is 1. It must be 1 for a liveness probe. · Failure Threshold: The minimum number of consecutive failed probes that are considered as failed after a successful probe. The default value is 3. The minimum value is 1.

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

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

d) Optional: Configure the lifecycle rule.

You can configure the following parameters for the container lifecycle: 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.

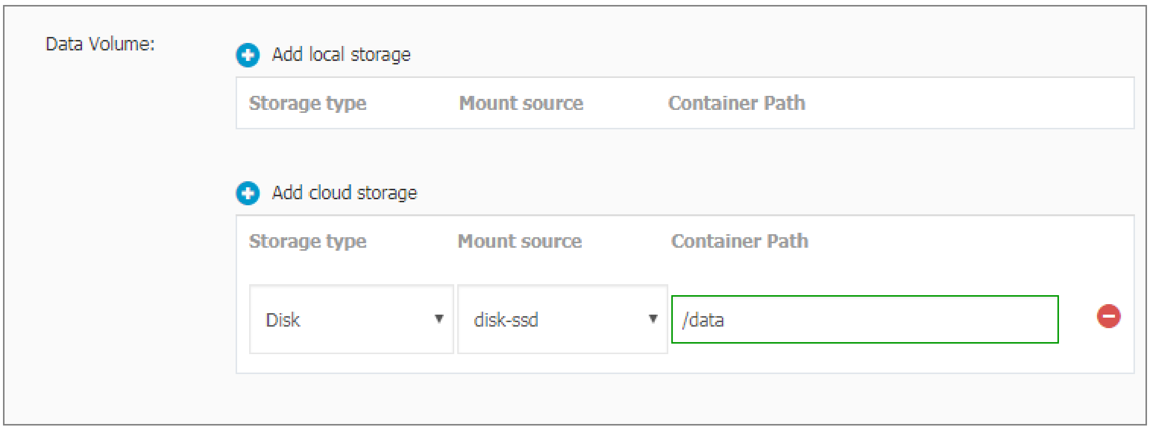
- Start: Configure a pre-start command and parameter for the container.
- Post Start: Configure a post-start command for the container.
- Pre Stop: Configure a pre-end command for the container.

e) Configure data volumes.

Local storage and cloud storage can be configured.

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

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



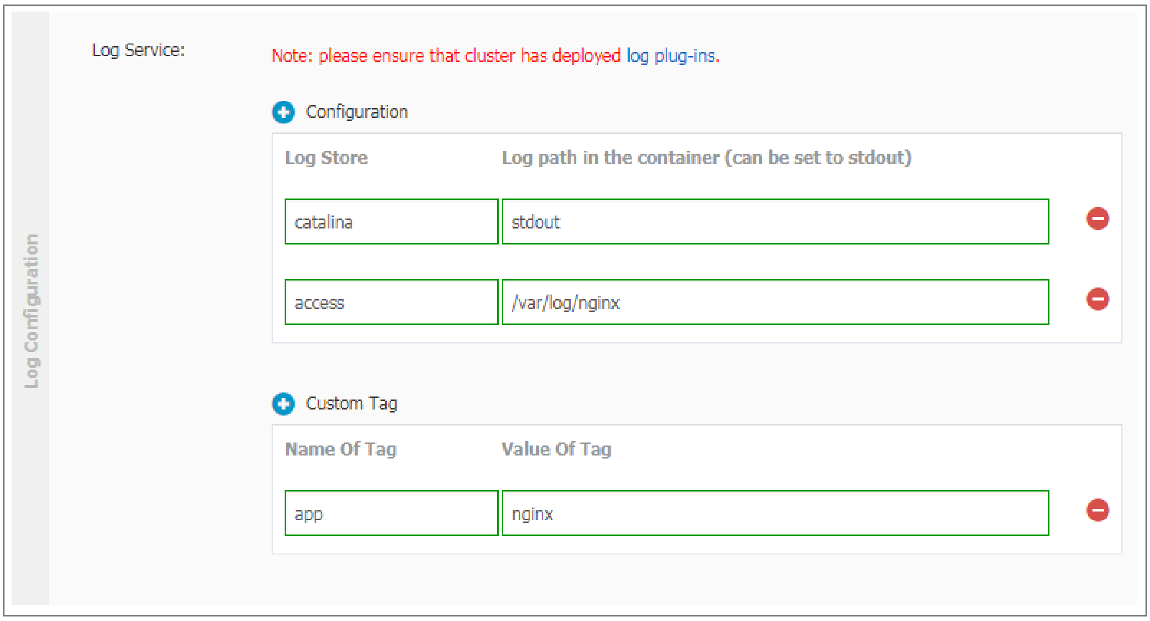
f) Optional: Configure Log Service. You can configure collection methods and customize tags for this service.

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

Configure log collection methods as follows:

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

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



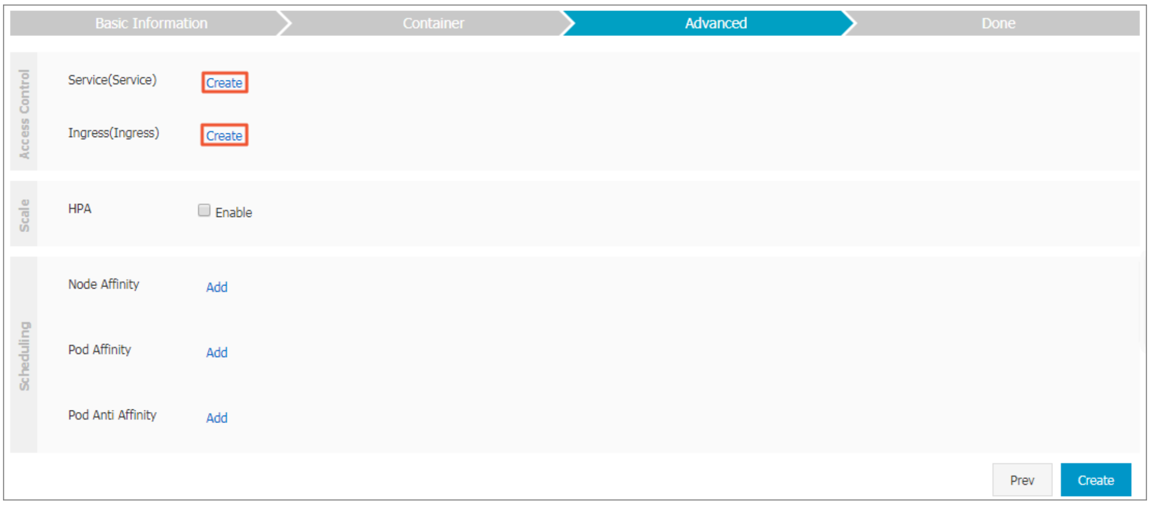
5. Click Next after completing the configurations.
6. Configure advanced settings. In this example, configure only access settings.
   a) Configure Access Control.

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.

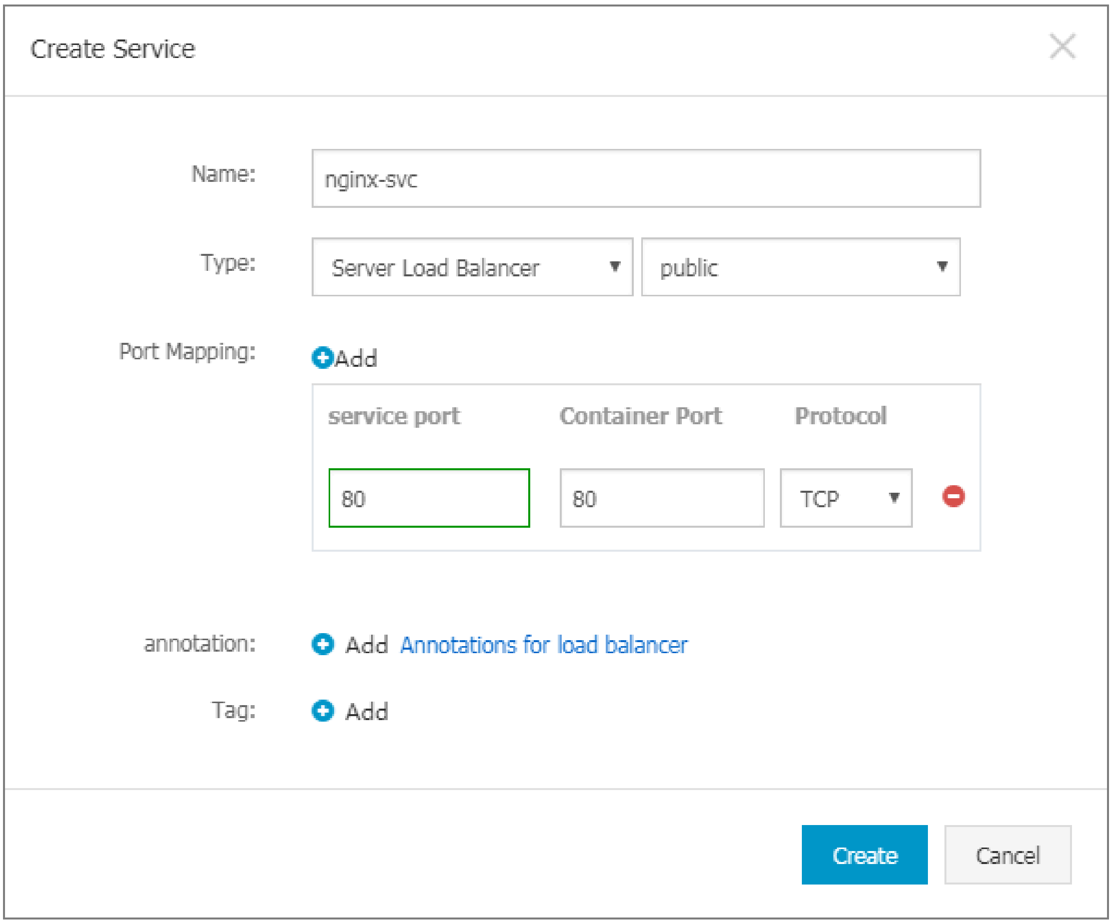
Note:

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/*.



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



- 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.

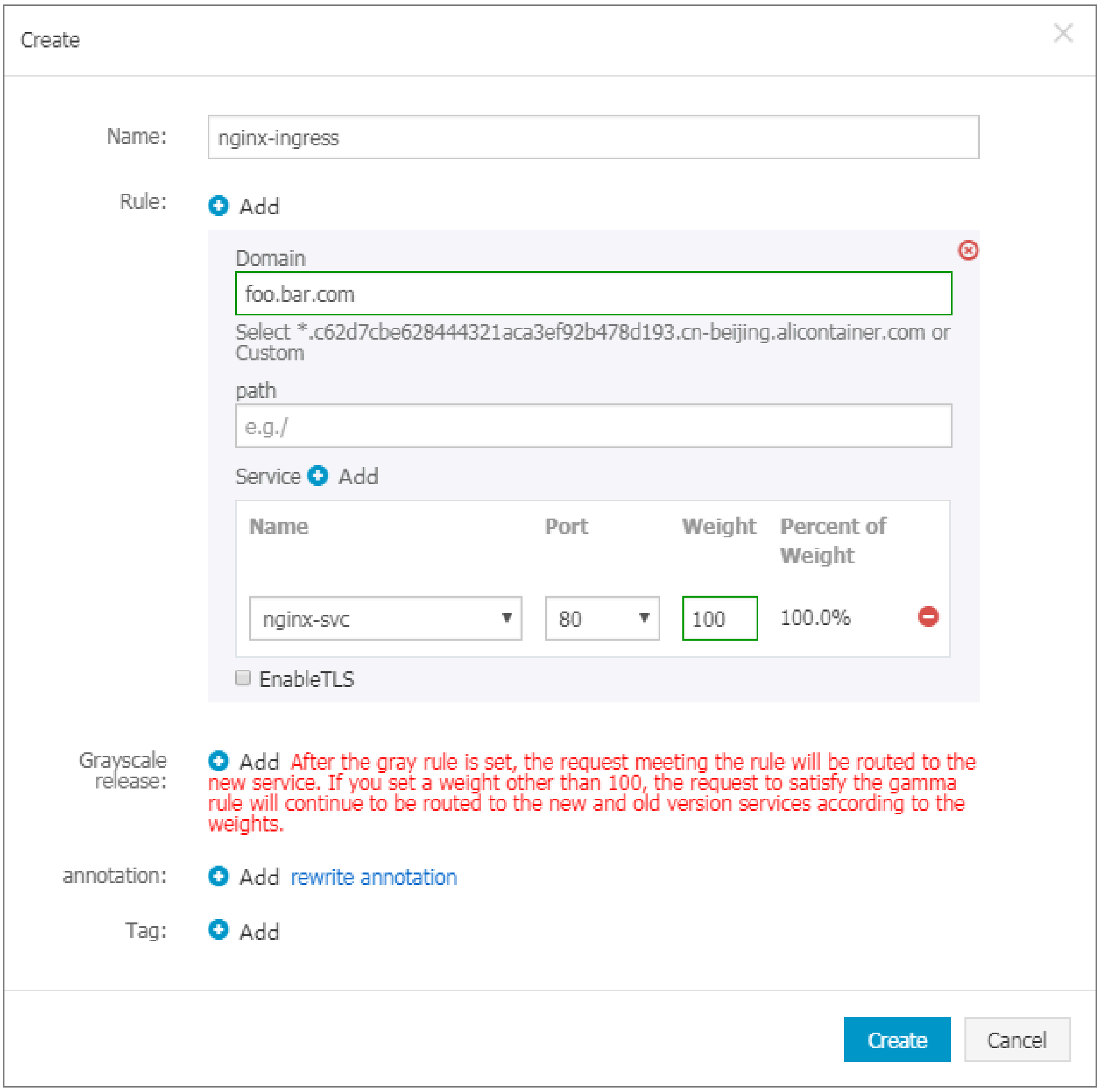
B. Click Create at the right of Ingress. Configure rout rules for the backend pod in the displayed dialog box, and then click Create. For more information about route configuration, see *Ingress configurations*.

Note:

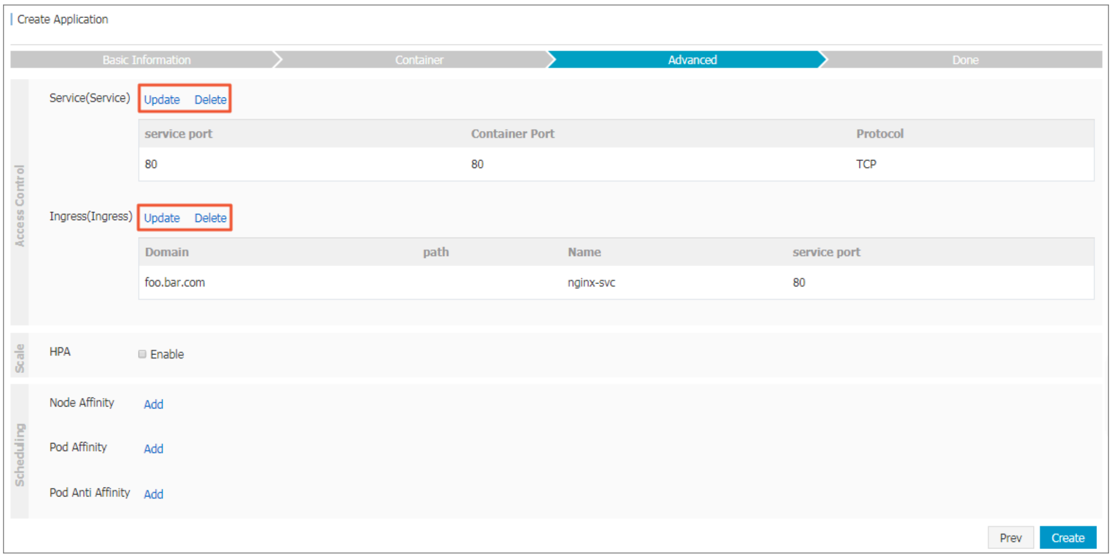
When you create an application by using an image, you can create ingress for only one service. In this example, use a virtual host name as the testing

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
```

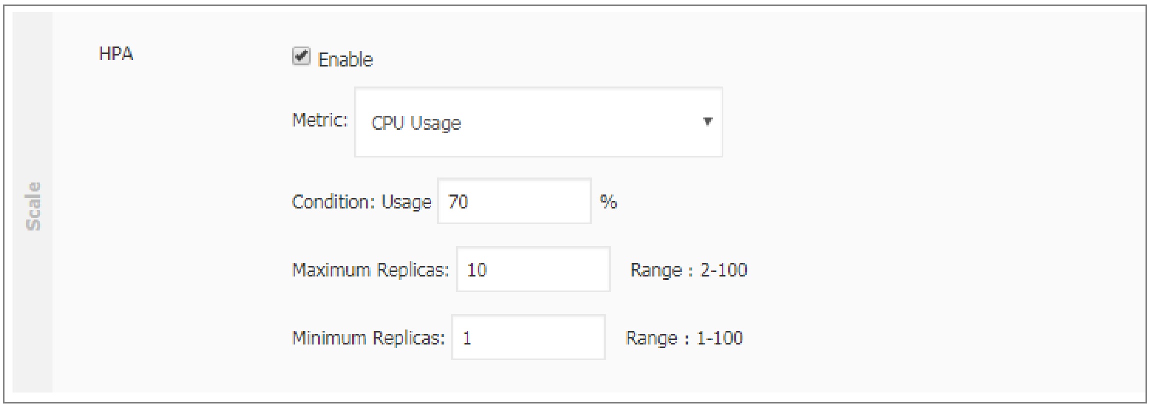


C. The created service and ingress are displayed in the access control section. You can reconfigure the service and ingress by clicking Update and Delete.



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.



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.

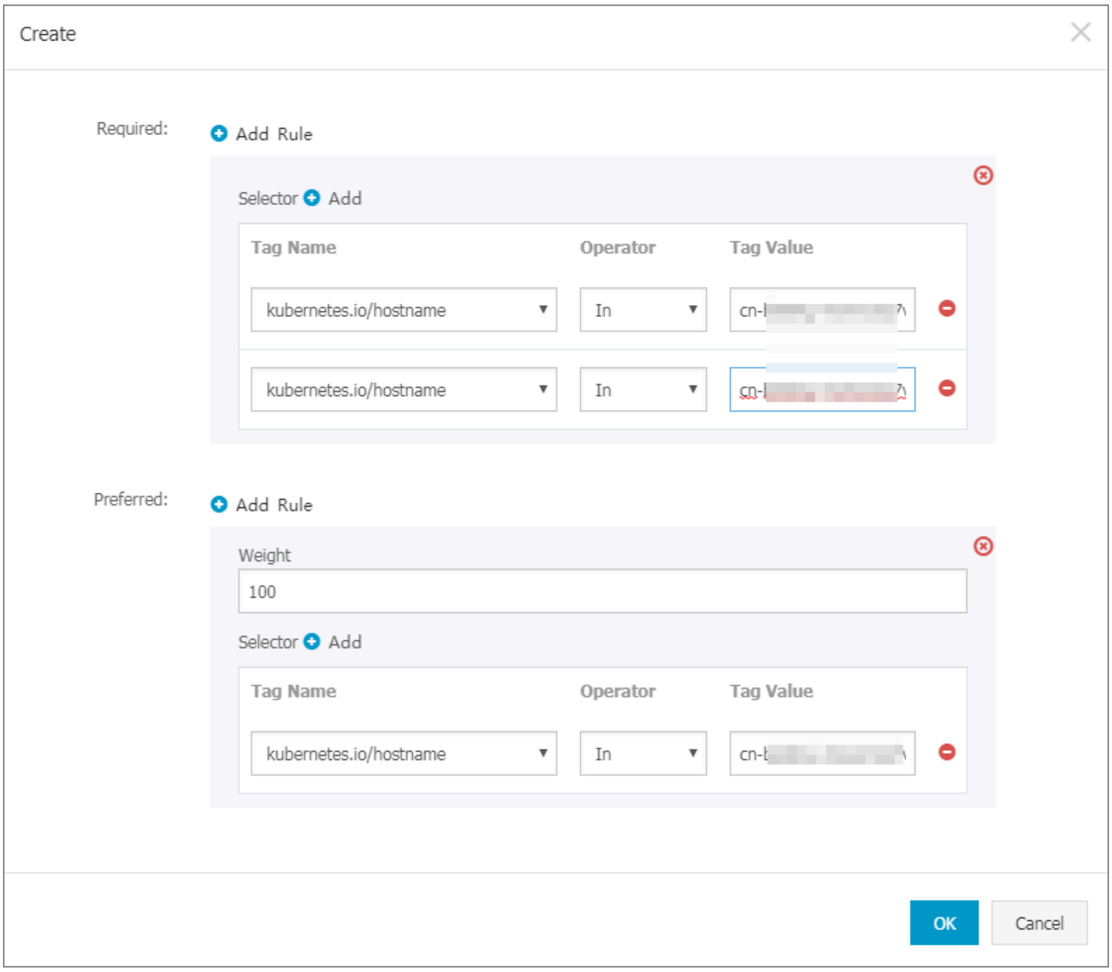
- Minimum Replicas: The minimum number of replicas that the deployment can contract to.

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.

A. Set Node Affinity by configuring node tags.

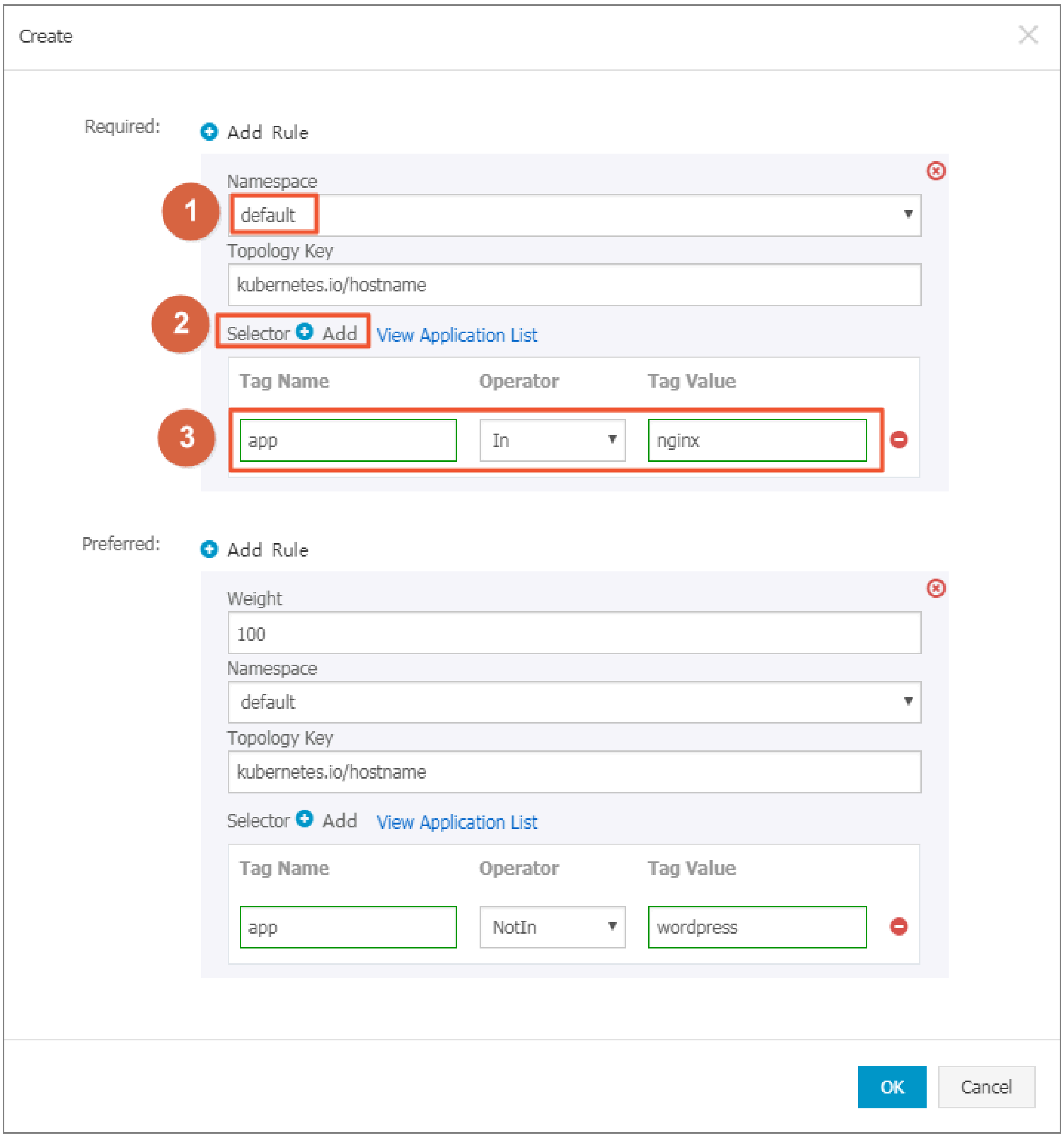


Node scheduling supports both required and preferred rules, and various operators such as In, NotIn, Exists, DoesNotExist, GT, and LT.

- 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.

B. 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.



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.

C. 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.

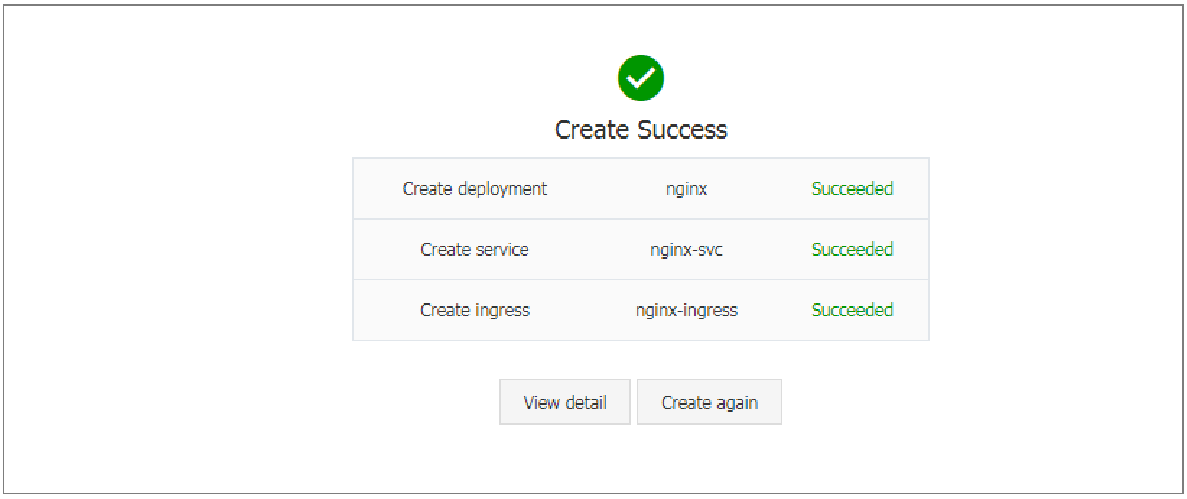
· Grant a pot the exclusive access to a node so as to guarantee no other pods use resources of the node.

· Distribute pods of services that may affect each other to different hosts.

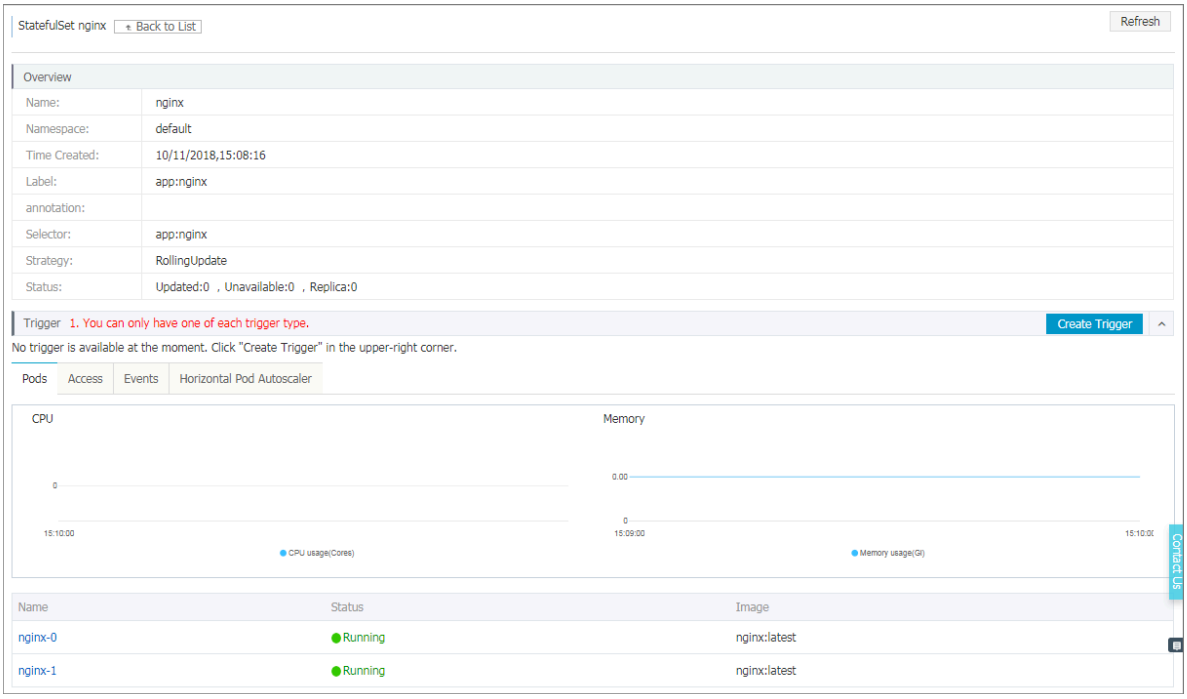
> Note:
> Configuration methods of pod anti affinity scheduling are the same as that of pod affinity. But the same scheduling rules have different meanings for

pod anti affinity scheduling. Select an appropriate scheduling rule based on scenarios.

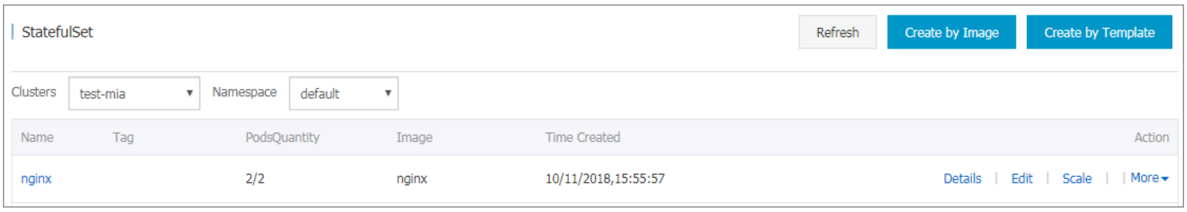
7. Click Create.
8. After you create the application, the create success page is displayed by default and objects contained in the application are listed. You can click View detail to view the deployment details.



The StatefulSet page is displayed by default.

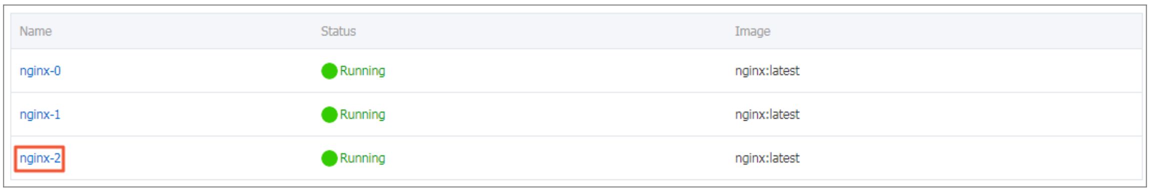


9. Then click Back to list in the upper-left corner to view the created StatefulSet application in the StatefulSet list page.

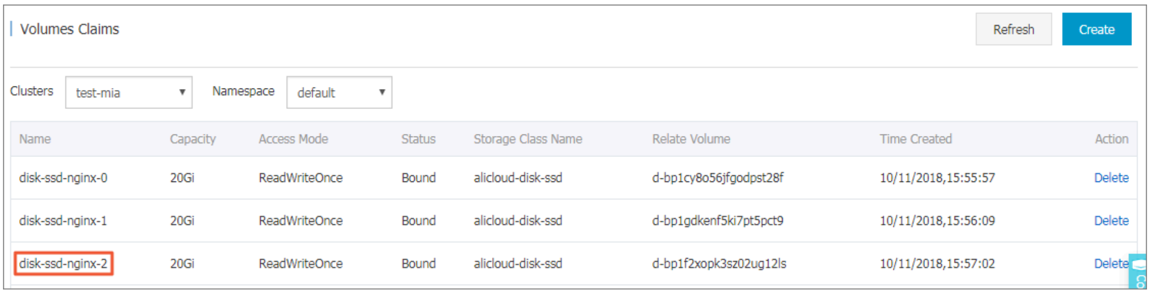


10.Optional: To verify service scalability, click Scale at the right of a target nginx application.

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



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



What's next

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

Create a temporary file on a cloud disk:

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

# kubectl exec nginx-1 touch /tmp/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.

1.7.3 Create a Job application by using an image

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

Prerequisites

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

Context

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

Kubernetes supports the following types of Jobs:

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

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

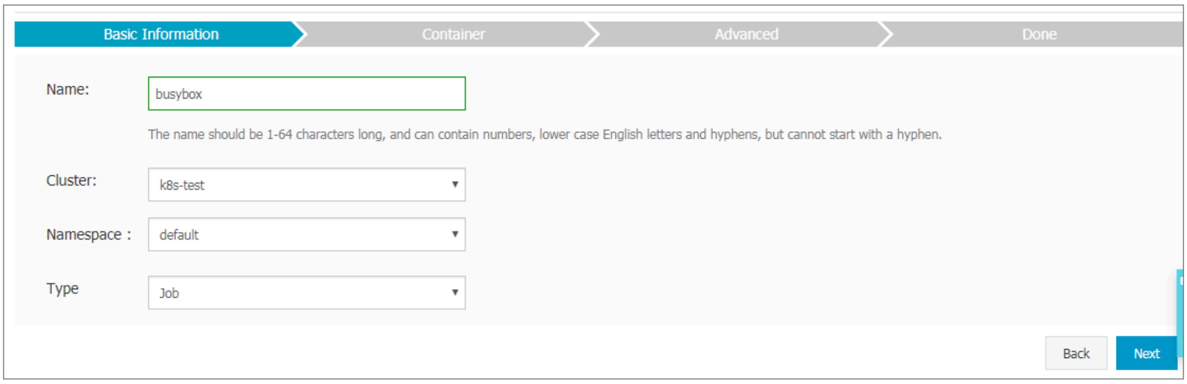
Note:
The Job created in this example is a parallel Job with a fixed completion count.

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

Procedure

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

     Note:
     In this example, select the Job type.



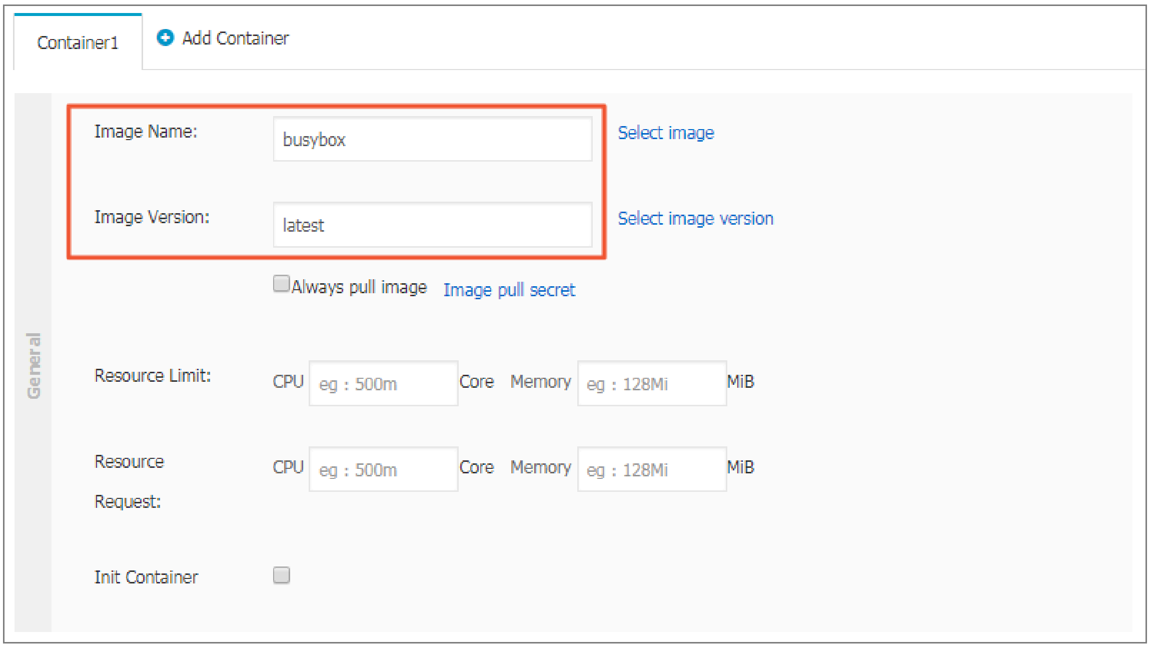
4. Configure containers.

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

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

        You can also enter a private registry in the format of `domainname/namespace/imagename:tag` to specify an image.
      - Image Version: Click Select image version to select a version. If you do not specify any image version, the system uses the latest version by default.

- Always pull image: Container Service caches the image to improve deployment efficiency. During deployment, if the tag of the newly specified image is the same as that of the cached image, Container Service reuses the cached image rather than pulls the same image again. Therefore, if you do not modify the image tag during scenarios where you are changing your code and image, the earlier image in the local cache is used in the application deployment. If you select this check box, Container Service ignores the cached image and re-pulls the image when deploying the application to make sure the latest image and code are always used.
- Image pull secret: If you use a private image, we recommend that you use a secret to guarantee the security of your image. For more information, see *Use an image secret*.
- Resource Limit: Specify the upper limit for the resources (CPU and memory) that can be used by this application to avoid occupying excessive resources. CPU is measured in millicores, that is, one thousandth of one core. Memory is measured in bytes, which can be Gi, Mi, or Ki.
- Resource Request: Specify how many resources (CPU and memory) are reserved for the application (that is, these resources become exclusive to the container). If you do not set this parameter, other services or processes will compete for resources, which means the application may become unavailable due to resource shortage.
- Init Container: Select this check box to create an Init Container that contains useful tools. For more information, see *https://kubernetes.io/docs/concepts/workloads/pods/init-containers/*.

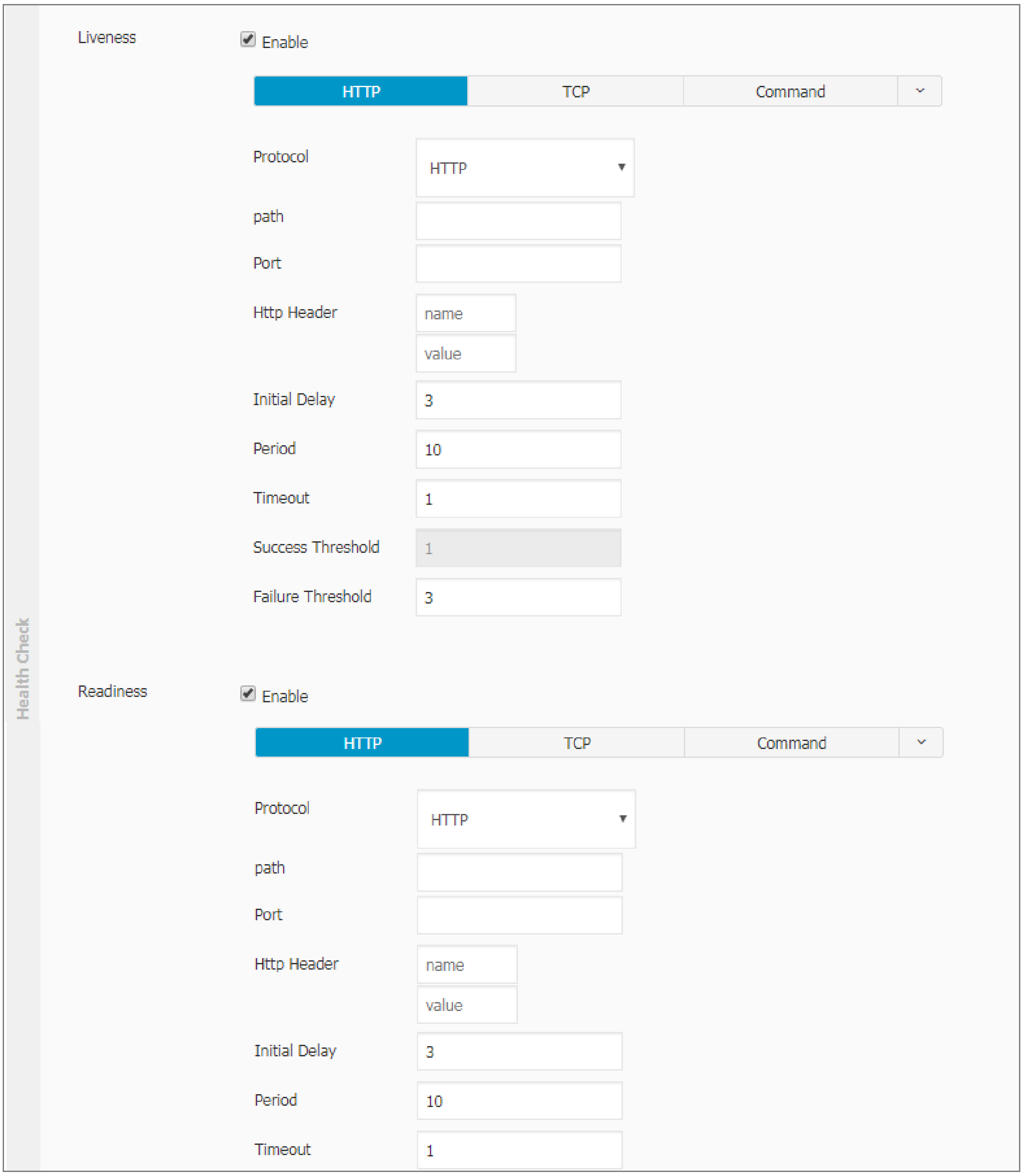


b) Optional: Set Environment .

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

c) Optional: Set Health Check.

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


Health Check
Liveness
Enable
HTTP
TCP
Command
Protocol
HTTP
path
Port
Http Header
name
value
Initial Delay
3
Period
10
Timeout
1
Success Threshold
1
Failure Threshold
3
Readiness
Enable
HTTP
TCP
Command
Protocol
HTTP
path
Port
Http Header
name
value
Initial Delay
3
Period
10
Timeout
1

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

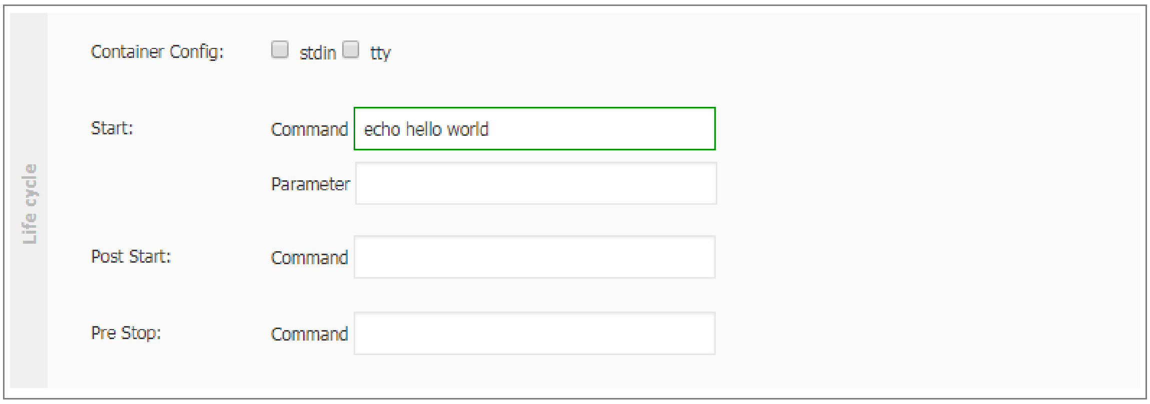
Request method	Description
TCP connection	If you use this health check method, a TCP socket is sent to the container. The kubelet then attempts to open the socket of the container on a specified port. If a connection can be established, the container is considered healthy. If not, it is considered unhealthy.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

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

d) Optional: Set the life cycle.

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

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



e) Optional: Set data volumes.

You can configure local storage and cloud storage.

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

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

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

Set the following log collection parameters:

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

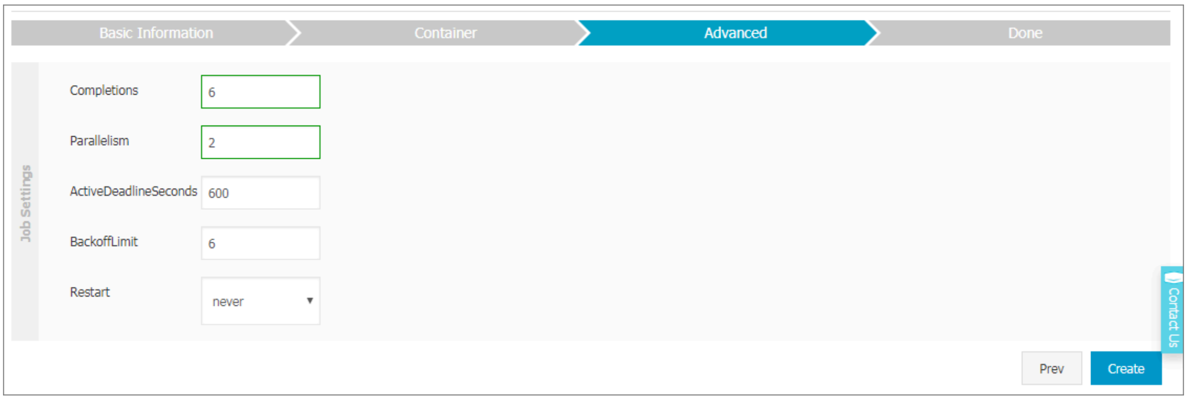
You can also set custom tags. The custom tags are collected to the container output logs. A custom tag can help you tag container logs, making it easy to collect log statistics, filter logs, and analyze logs by using other methods.

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

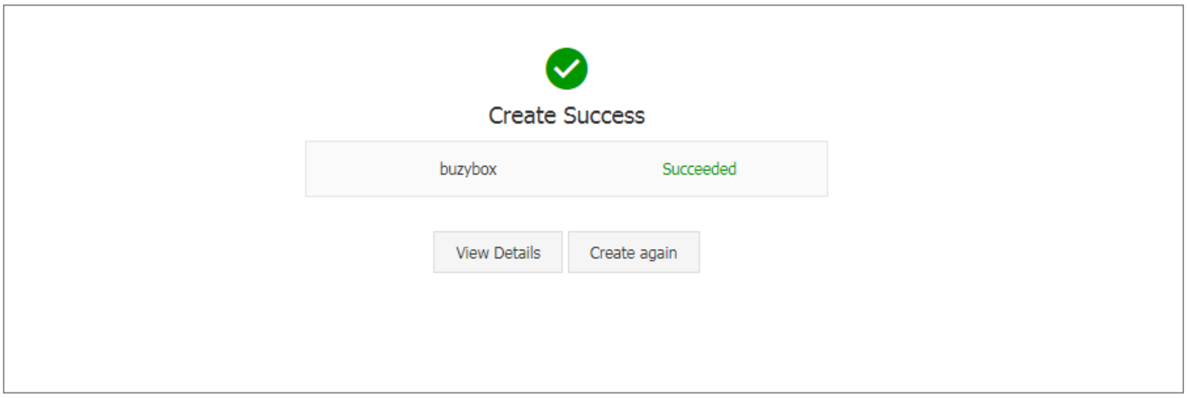
   You can configure Job Settings.

Parameter	Description
Completions	Number of pods that must be run successfully by the configured Job. The default value is 1.
Parallelism	Number of pods that must be run in parallel by the configured Job at any time. The default value is 1.
ActiveDeadlineSeconds	Operating time limit of the configured Job. If the Job is not completed within the time limit, the system tries to terminate the Job.
BackoffLimit	Number of retries performed by the configured Job to create pods after a failure. The default is 6. Each time the Job fails, the failed pods associated with the Job are recreated with time delay . The time delay grows exponentially each time. The upper limit of the time delay is six minutes.

Parameter	Description
Restart	Only Never and OnFailure restart policies are supported.

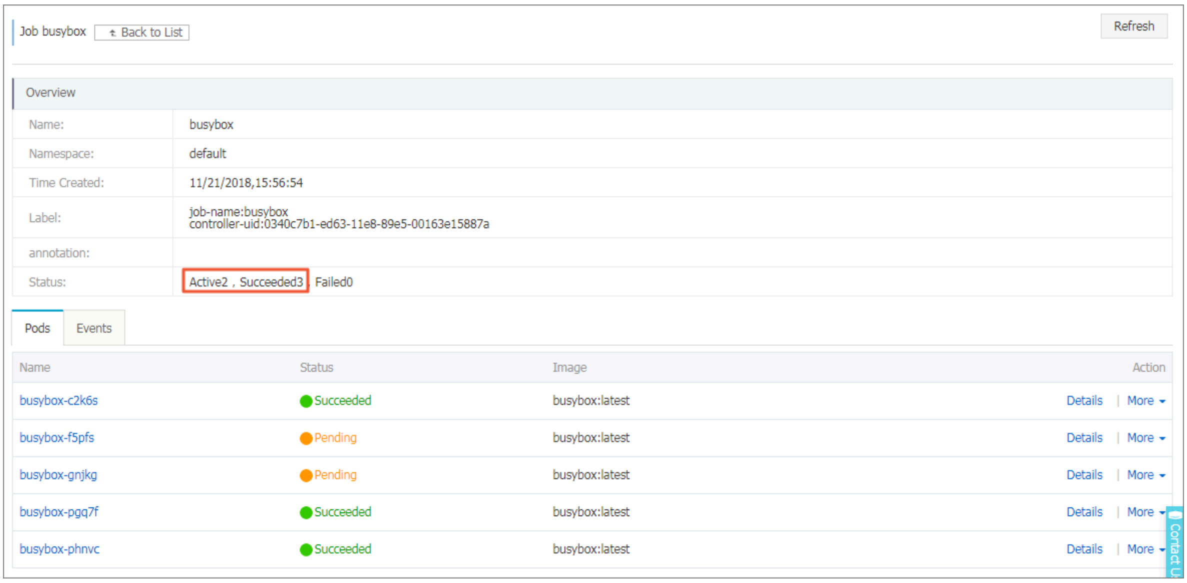


7. Click Create.
8. After you create the Job application, a new page is displayed by default to prompt that you have created the application with the objects included.

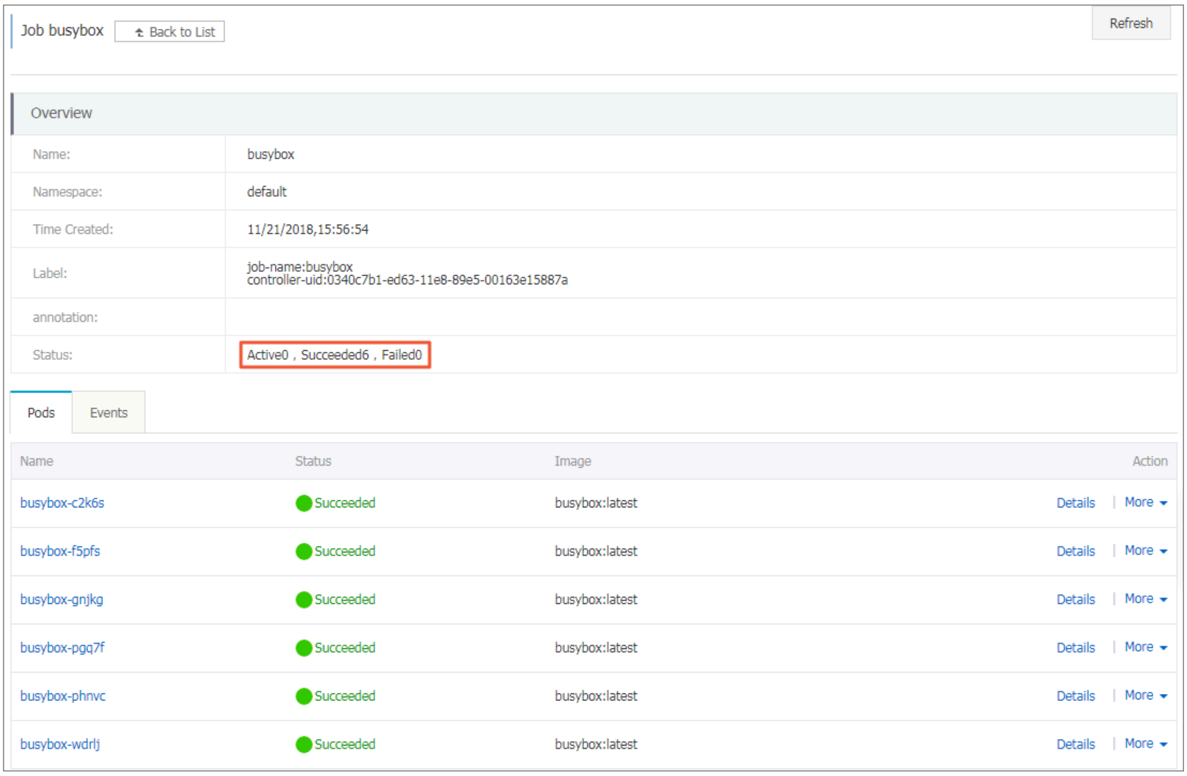


You can click View Details to view the Job details.

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

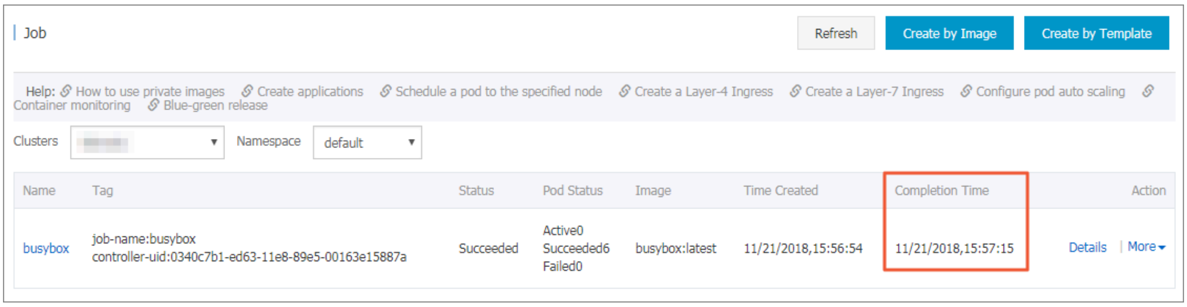


Wait until all pods are created.



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

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



1.7.4 Create an application in Kubernetes dashboard

You can create an application in the Kubernetes dashboard.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click Clusters in the left-side navigation pane.
3. Click Dashboard at the right of the cluster to enter the Kubernetes dashboard.
4. In the Kubernetes dashboard, click CREATE in the upper-right corner to create an application.
5. The Resource creation page appears. Configure the application information.

   Create an application in any of the following three ways:

   - CREATE FROM TEXT INPUT: Directly enter the orchestration codes in the YAML or JSON format to create an application. You must know the corresponding orchestration format.
   - CREATE AN APP: Complete the following configurations to create an application.
     - App name: Enter the name of the application you are about to create. In this example, enter `nginx-test`.
     - Container image: Enter the URL of the image to be used. In this example, use Docker *Nginx*.
     - Number of pods: Configure the number of pods for this application.

- Service: Select External or Internal. External indicates to create a service that can be accessed from outside the cluster. Internal indicates to create a service that can be accessed from within the cluster.
- Advanced options: To configure the information such as labels and environment variables, click SHOW ADVANCED OPTIONS. This configuration distributes the traffic load evenly to three pods.

· CREATE FROM FILE: Upload an existing YAML or JSON configuration file to create an application.

6. Click UPLOAD or DEPLOY to deploy the containers and services.

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

What's next

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

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

1.7.5 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*.

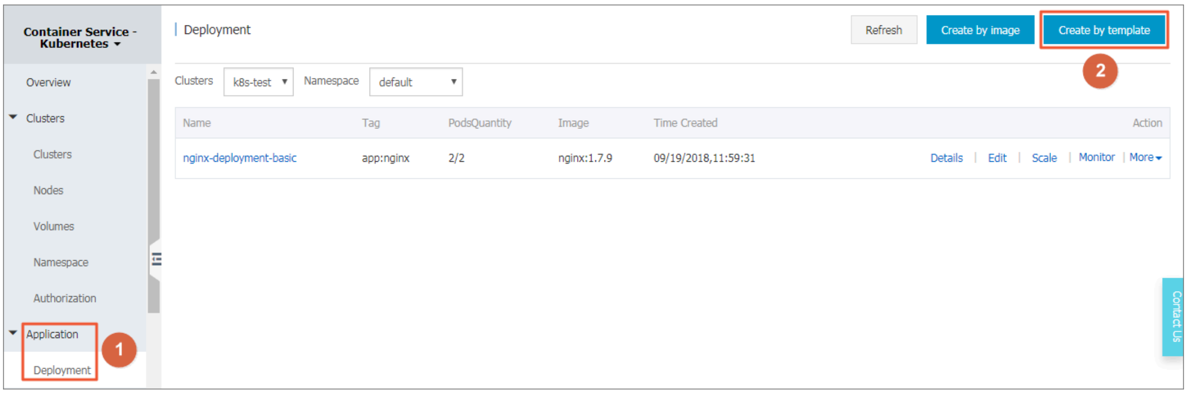
Context

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

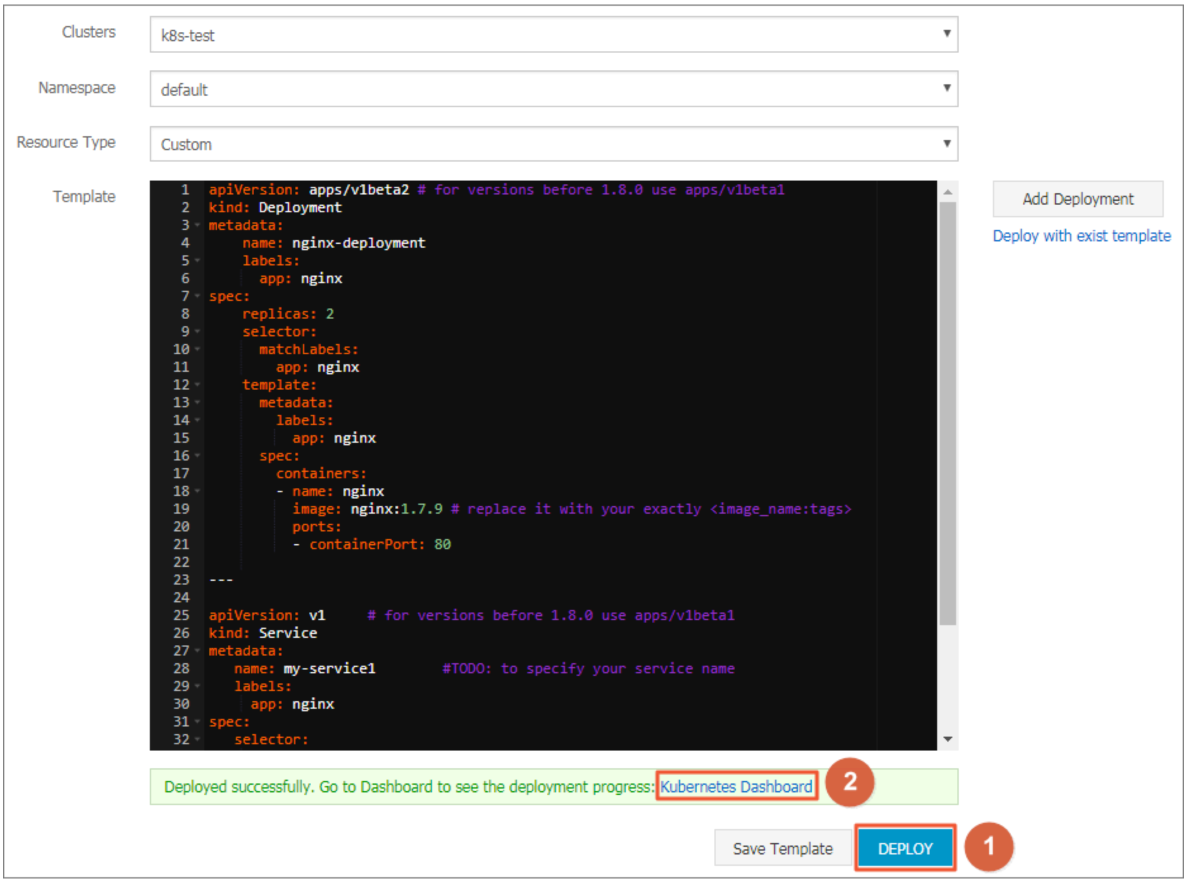
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Application > Deployment in the left-side navigation pane.

3. Click Create by template in the upper-right corner.



4. Configure the template and then click DEPLOY.

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



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

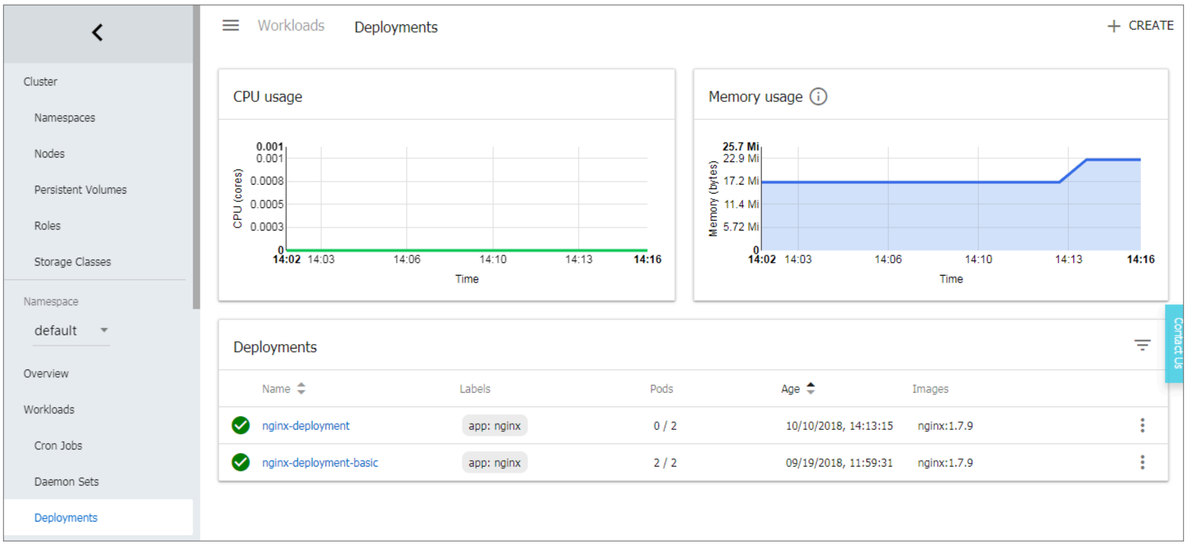
Note:

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

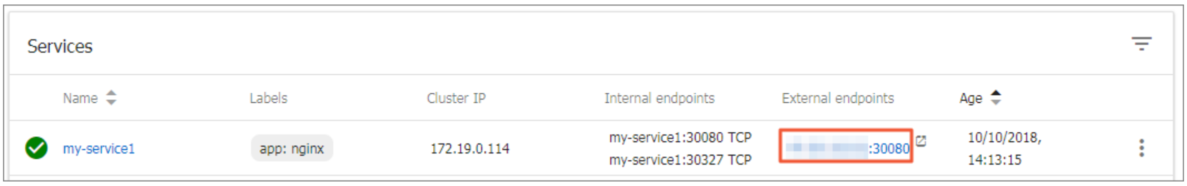
```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/
v1beta1
kind: Deployment
metadata:
    name: nginx-deployment
    labels:
      app: nginx
spec:
    replicas: 2
    selector:
      matchLabels:
        app: nginx
    template:
      metadata:
        labels:
```

```
          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.



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



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

What's next

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

1.7.6 Pull a private image without a password

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

Prerequisites

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

Context

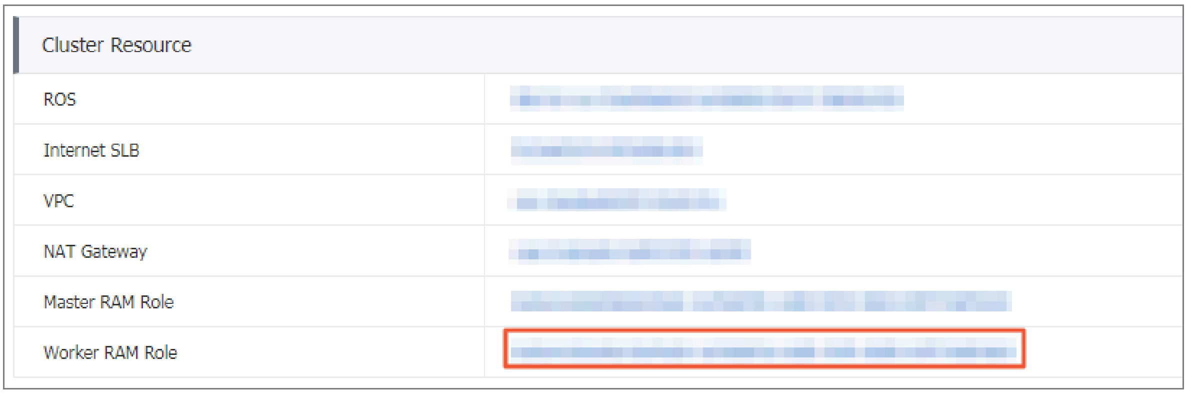
Function overview

- You can only pull a private image from an Alibaba Cloud container image repository that belongs to your account.
- You can pull a private image from a cross-region Alibaba Cloud container image repository.
- You can only perform this operation in the default namespace.
- Kubernetes clusters that support this function include:
  - Dedicated Kubernetes clusters
  - Managed Kubernetes clusters
  - Serverless Kubernetes clusters
- The following are Kubernetes cluster versions that support this function:

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

Procedure

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



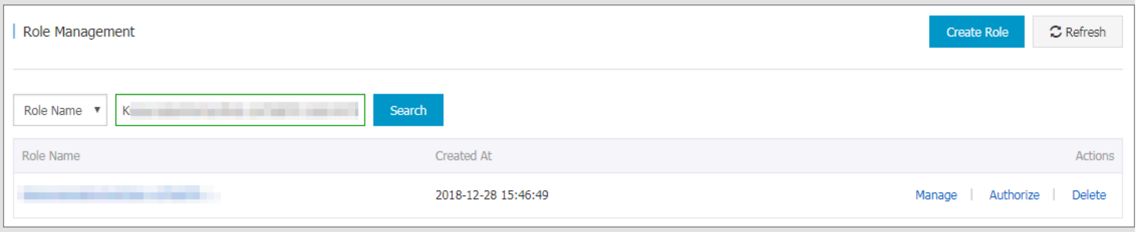
Note:

This topic uses the latest version of the RAM console.

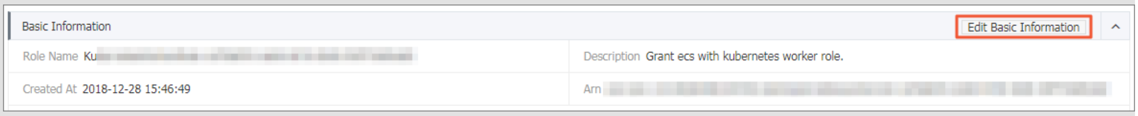
If you use an earlier version of the RAM console, you can modify the target policy document by using either of the following two methods:

Method 1

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

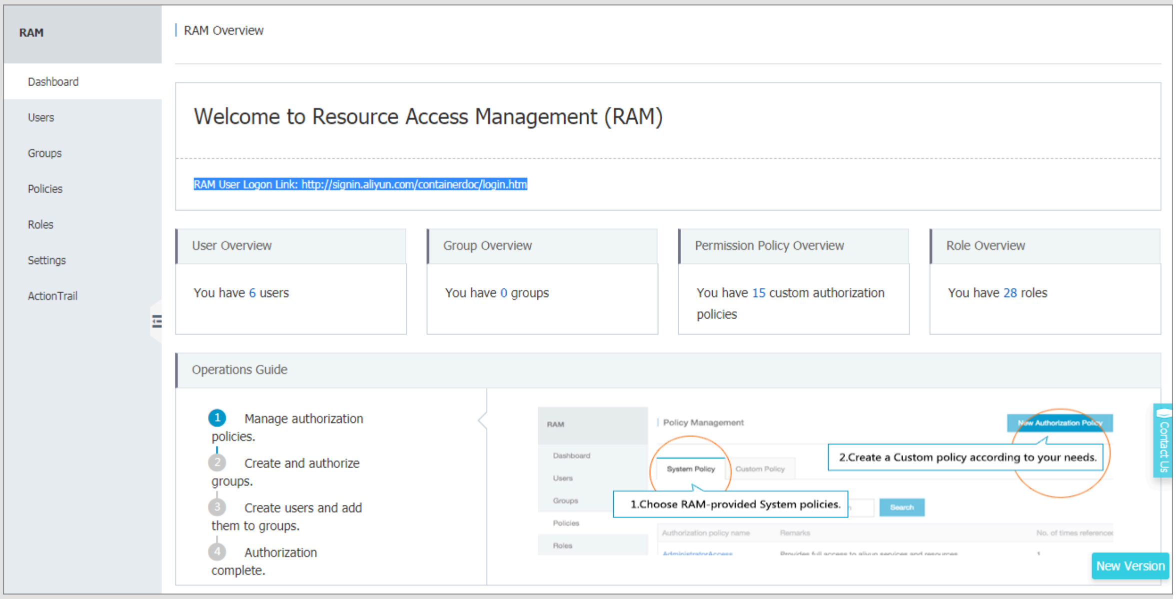


b. In the Basic Information area, click Edit Basic Information in the upper-right corner.

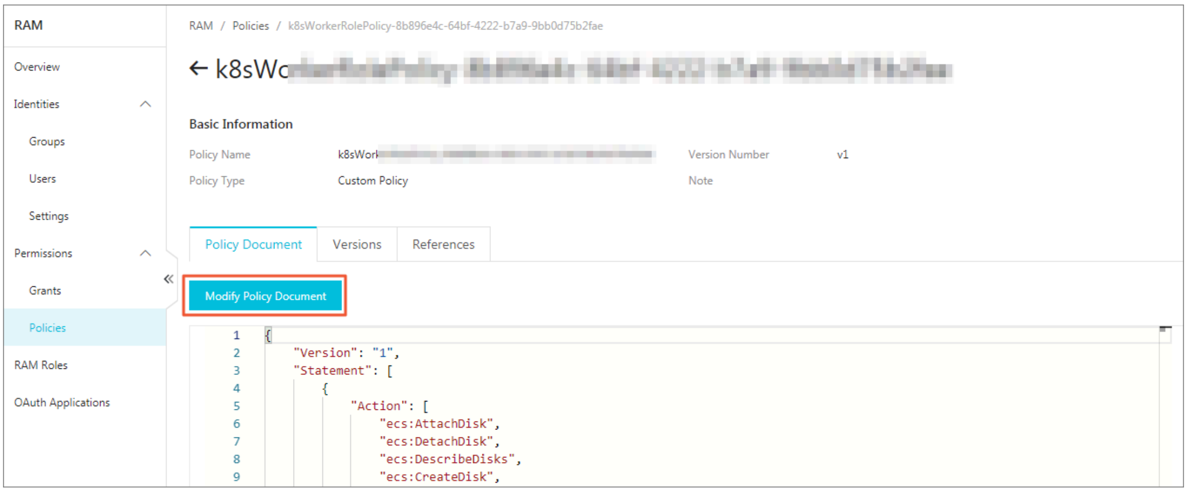


Method 2

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



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



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

```
{
            "Action": [
              "cr:Get*",
              "cr:List*",
```

```
                "cr:PullRepository"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
```

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

```
apiVersion: v1
kind: ServiceAccount
```

```
metadata:
     name: aliyun-acr-credential-helper
     namespace: kube-system
 ---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
     name: aliyun-acr-credential-helper-rolebinding
     namespace: kube-system
roleRef:
     apiGroup: rbac.authorization.k8s.io
     kind: ClusterRole
     name: cluster-admin
subjects:
     - kind: ServiceAccount
       name: aliyun-acr-credential-helper
       namespace: kube-system
 ---
 #kubectl create secret docker-registry acr-image-pull-secret-public
 --docker-server=cr-tmp-xxx --docker-username=cr-temp-xxx --docker-
password=cr-temp-xxx --docker-email=cr-temp-xxx
apiVersion: v1
data:
     .dockerconfigjson: eyJhdXRocyI6eyJjci10bXAteHh4Ijp7InVzZXJu
YW1lIjoiY3ItdGVtcC14eHgiLCJwYXNzd29yZCI6ImNyLXRlbXAteHh4Iiwi
ZW1haWwiOiJjci10ZW1wLXh4eCIsImF1dGgiOiJZM0l0ZEdWdGNDMTRlSGc2
WTNJdGRHVnRjQzE0ZUhnPSJ9fX0=
kind: Secret
metadata:
     name: aliyun-acr-credential-a
     namespace: default
type: kubernetes.io/dockerconfigjson
 ---
 #kubectl create secret docker-registry acr-image-pull-secret-vpc
 --docker-server=cr-tmp-xxx --docker-username=cr-temp-xxx --docker-
password=cr-temp-xxx --docker-email=cr-temp-xxx
apiVersion: v1
data:
     .dockerconfigjson: eyJhdXRocyI6eyJjci10bXAteHh4Ijp7InVzZXJu
YW1lIjoiY3ItdGVtcC14eHgiLCJwYXNzd29yZCI6ImNyLXRlbXAteHh4Iiwi
ZW1haWwiOiJjci10ZW1wLXh4eCIsImF1dGgiOiJZM0l0ZEdWdGNDMTRlSGc2
WTNJdGRHVnRjQzE0ZUhnPSJ9fX0=
kind: Secret
metadata:
     name: aliyun-acr-credential-b
     namespace: default
type: kubernetes.io/dockerconfigjson
 ---
apiVersion: apps/v1beta2
kind: Deployment
metadata:
     name: aliyun-acr-credential-helper
     namespace: kube-system
     labels:
       app: aliyun-acr-credential-helper
spec:
     replicas: 1
     selector:
       matchLabels:
         app: aliyun-acr-credential-helper
     template:
       metadata:
         labels:
           app: aliyun-acr-credential-helper
```

```
        spec:
          serviceAccount: aliyun-acr-credential-helper
          containers:
          - name: aliyun-acr-credential-helper
            image: registry.cn-shanghai.aliyuncs.com/acs/aliyun-acr-
credential-helper:v18.10.29.0-1a28f02-aliyun
            imagePullPolicy: Always
          terminationGracePeriodSeconds: 0
```

1.7.7 Manage applications by using commands

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

Prerequisites

Before using commands to manage applications, *#unique_56*.

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
```

1.7.8 Simplify Kubernetes application deployment by using Helm

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

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

This document introduces the basic concepts and usage of Helm and 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

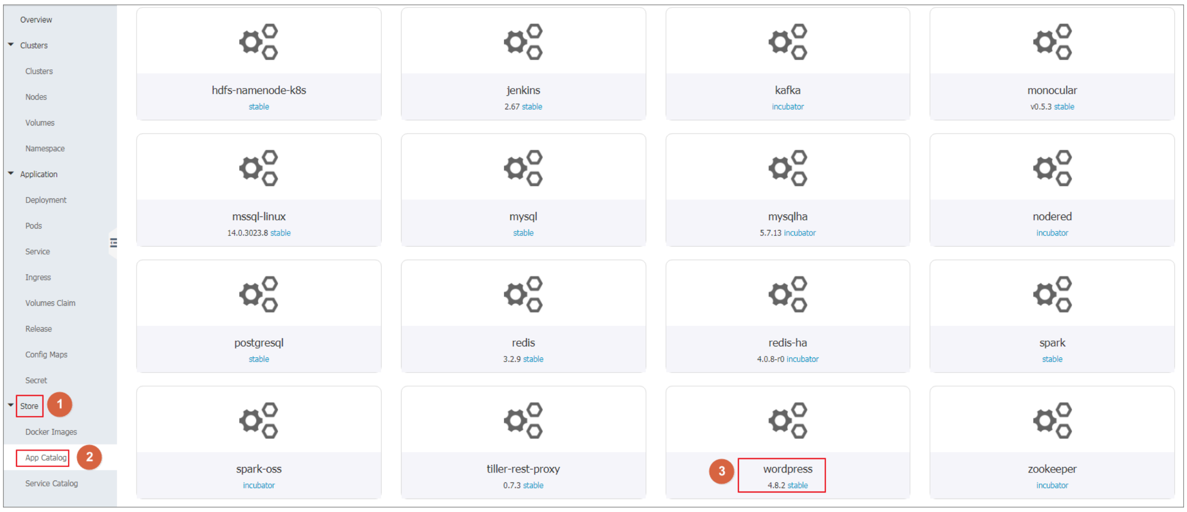
- Before using Helm to deploy an application, create a Kubernetes cluster in Alibaba Cloud Container Service. For more information, see *Create a Kubernetes cluster*.

  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.

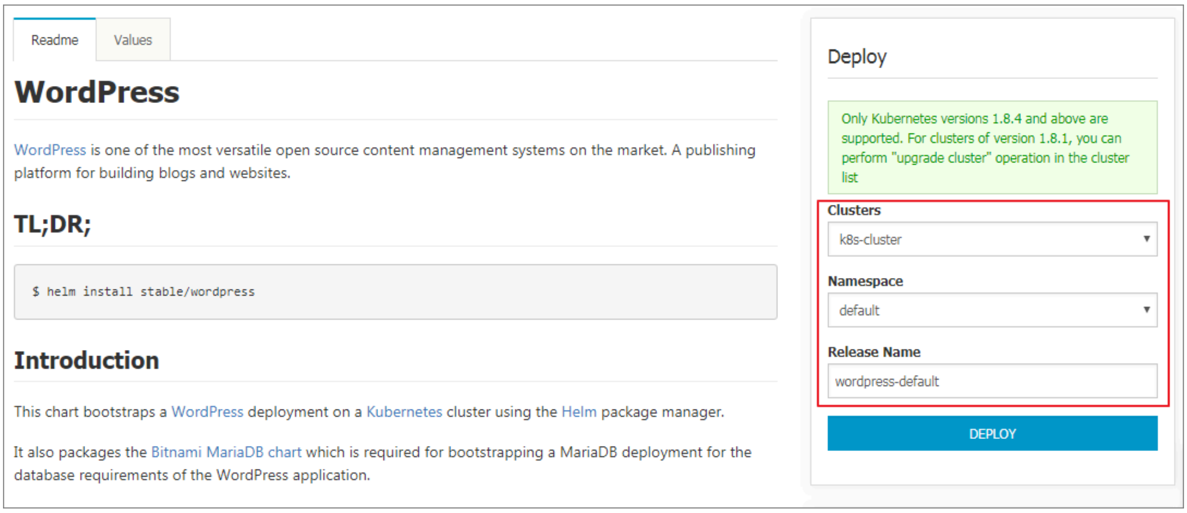
  Only clusters whose Kubernetes version is 1.8.4 or later are supported. For clusters whose Kubernetes version is 1.8.1, upgrade the cluster on the Cluster List page.

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.

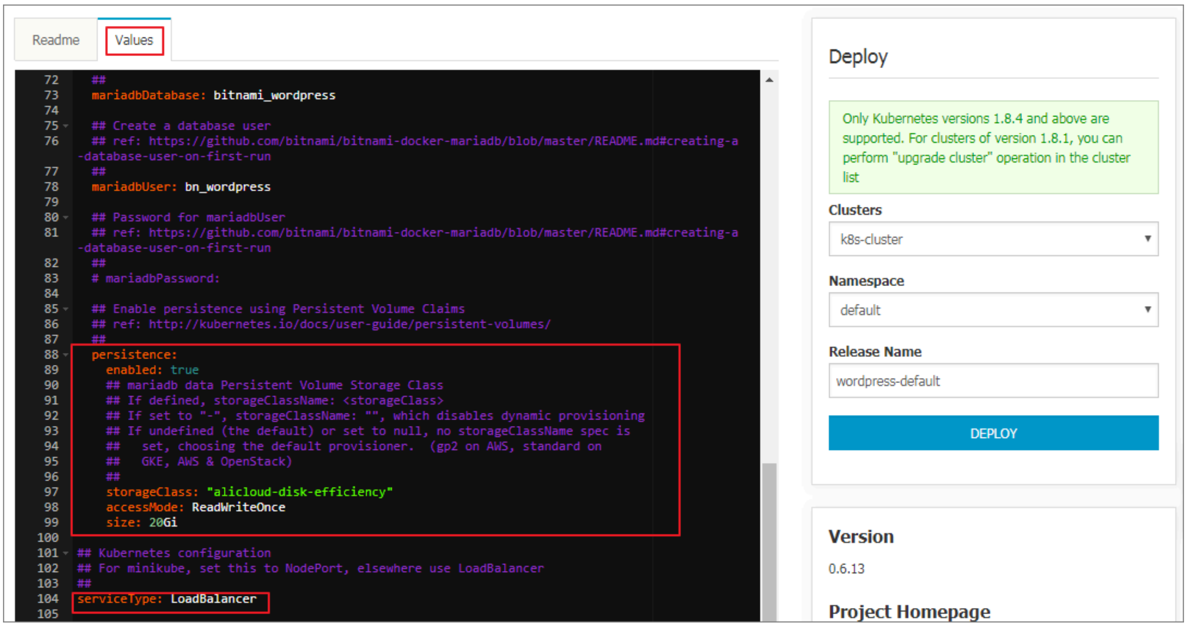


5. Click the Values tab to modify the configurations.

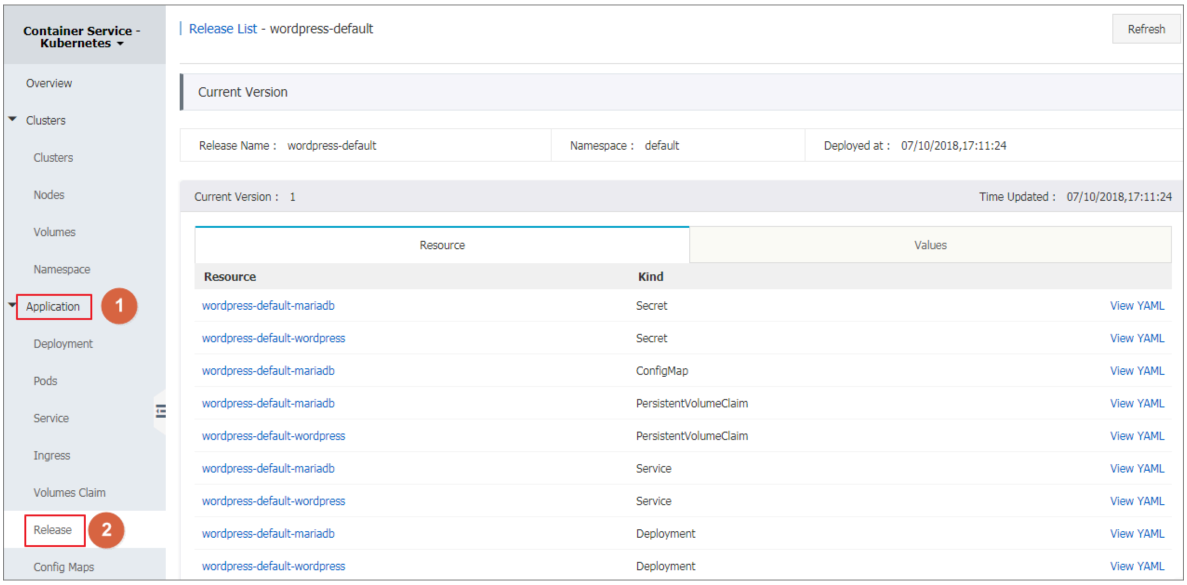
   In this example, bind dynamic data volumes of the cloud disk to a persistent storage volume claim (PVC). For more information, see .

   Note:

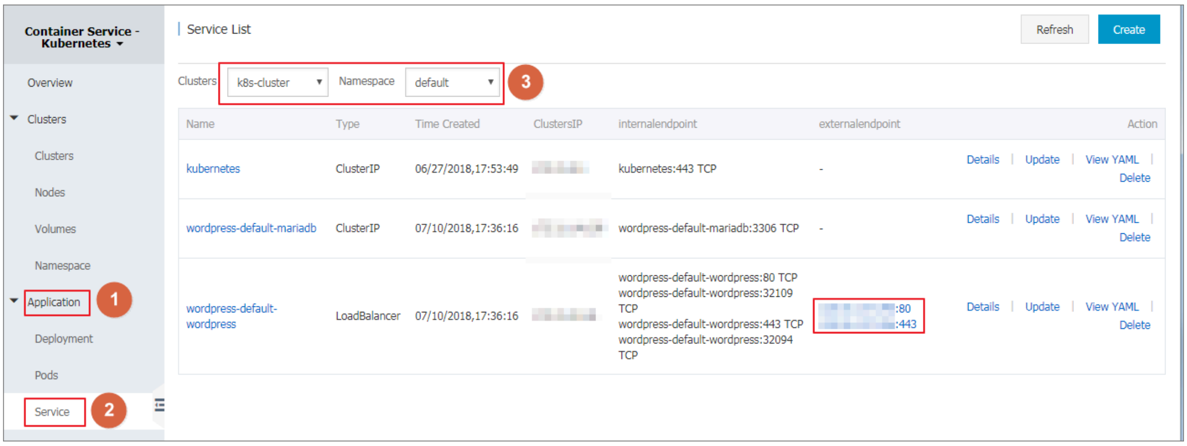
   You need to create a persistent storage volume (PV) of cloud disk in advance. The capacity of the PV cannot be less than the value defined by the PVC.



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



7. Click 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.



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/`.

1.7.9 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 availability. 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 implementations of the backend.

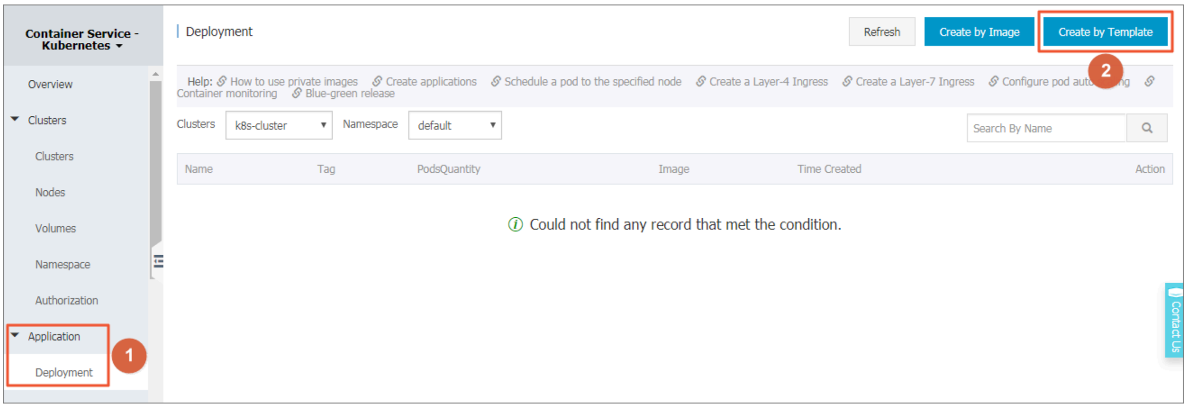
For more information, see *Kubernetes service*.

Prerequisite

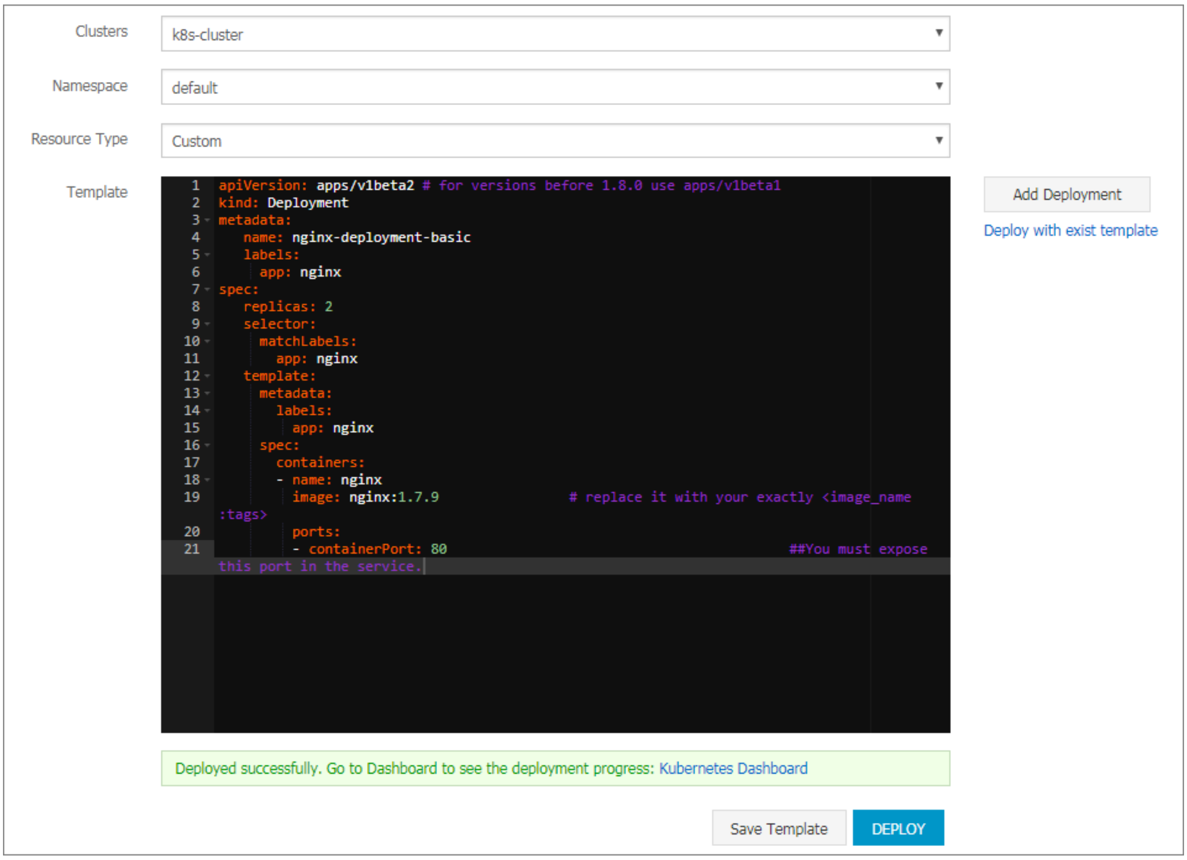
You have created a Kubernetes cluster successfully. For how to create a Kubernetes cluster, see *#unique_40*.

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.



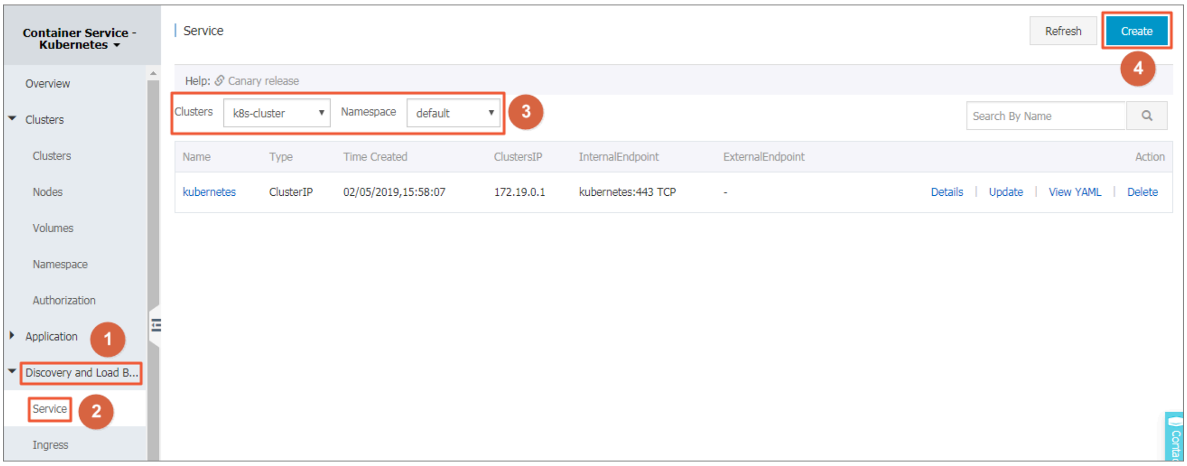
In this example, the sample template is an Nginx deployment.

```
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.
```

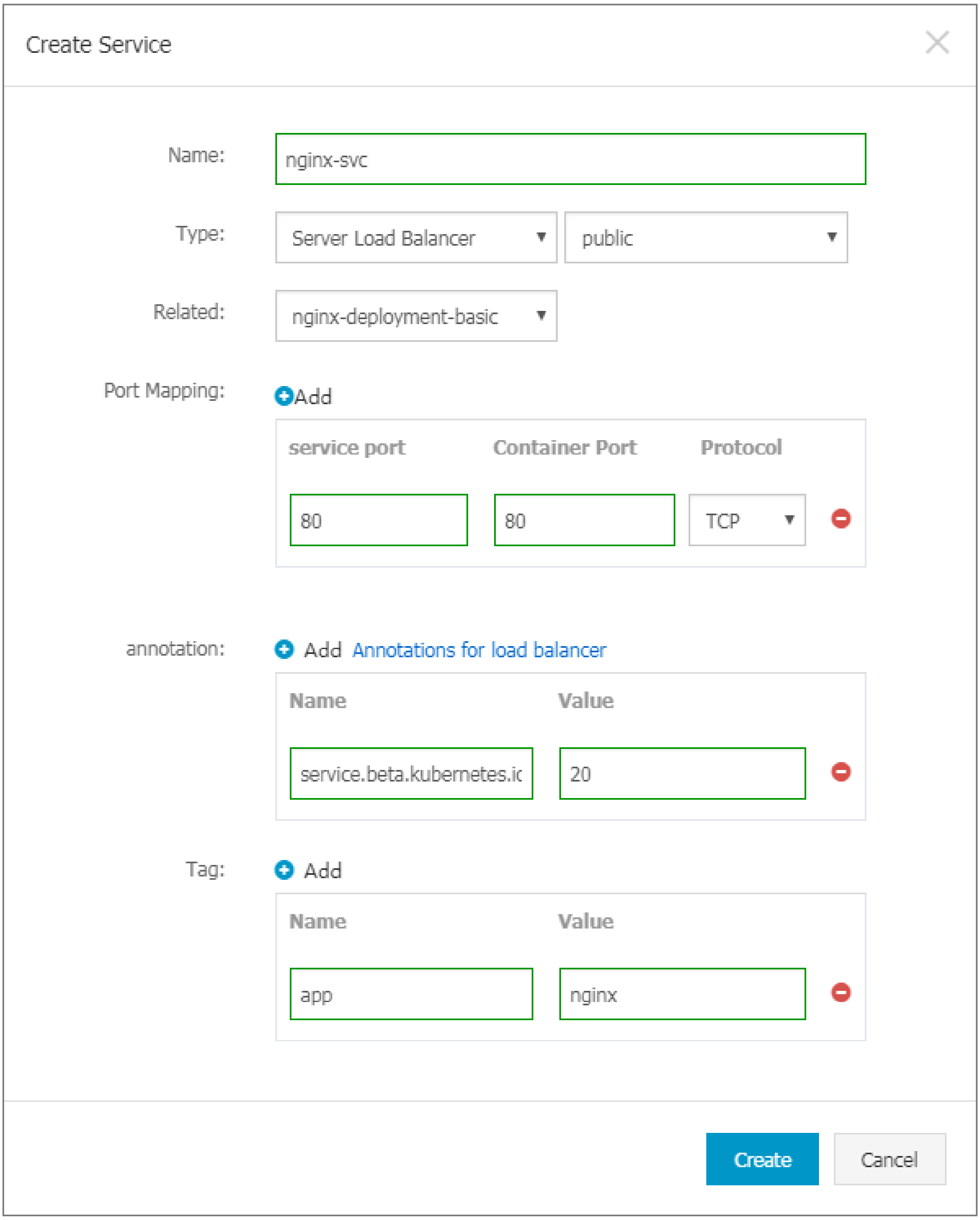
4. Go to the Kubernetes dashboard to view the running status of this deployment.

Step 2 Create a service

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, choose Discovery and Load Balancing > Service .
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click Create in the upper-right corner.

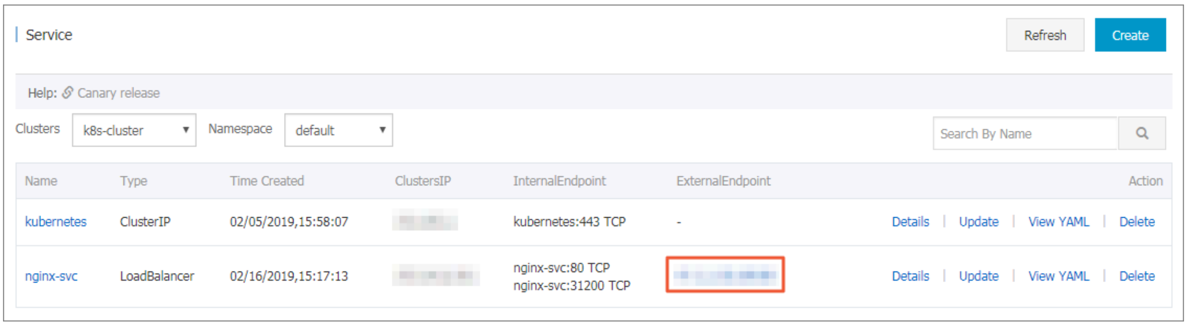


4. Complete the configurations in the displayed Create Service dialog box.



- 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.
  - annotation: Add an annotation to the service. You can set SLB parameters. For example, to set the peak bandwidth of the service to 20 Mbit/s, you can set this parameter as `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20`. For more information, see *Access services by using Server Load Balancer*.
  - Tag: Add a tag to the service to identify the service.
5. Click Create. The nginx-svc service is 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.

1.7.10 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.

1.7.11 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 > > Servicein 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.

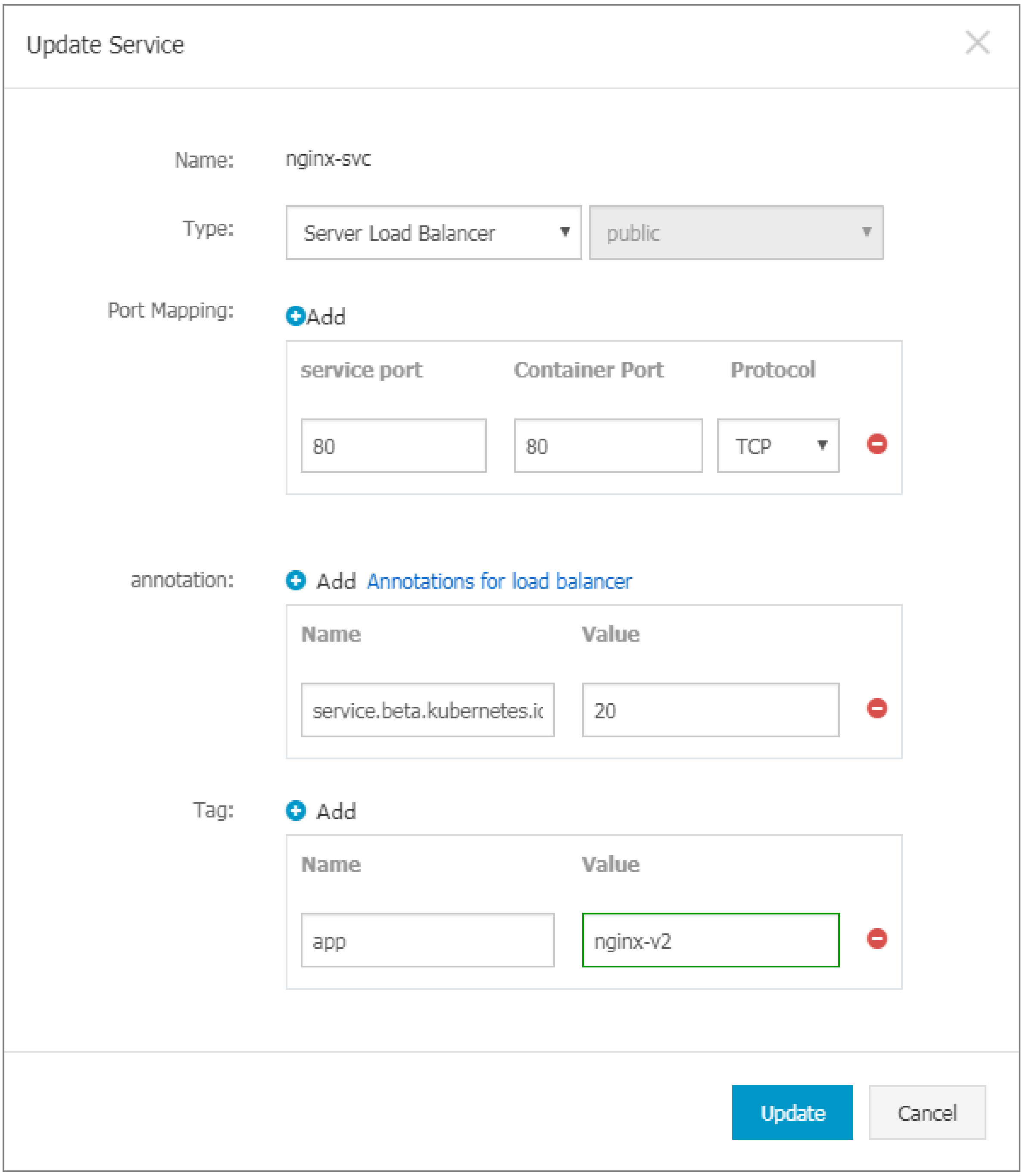
   You can also enter the Kubernetes dashboard of the cluster and click Services in the left-side navigation pane to view the services.

1.7.12 Update a service

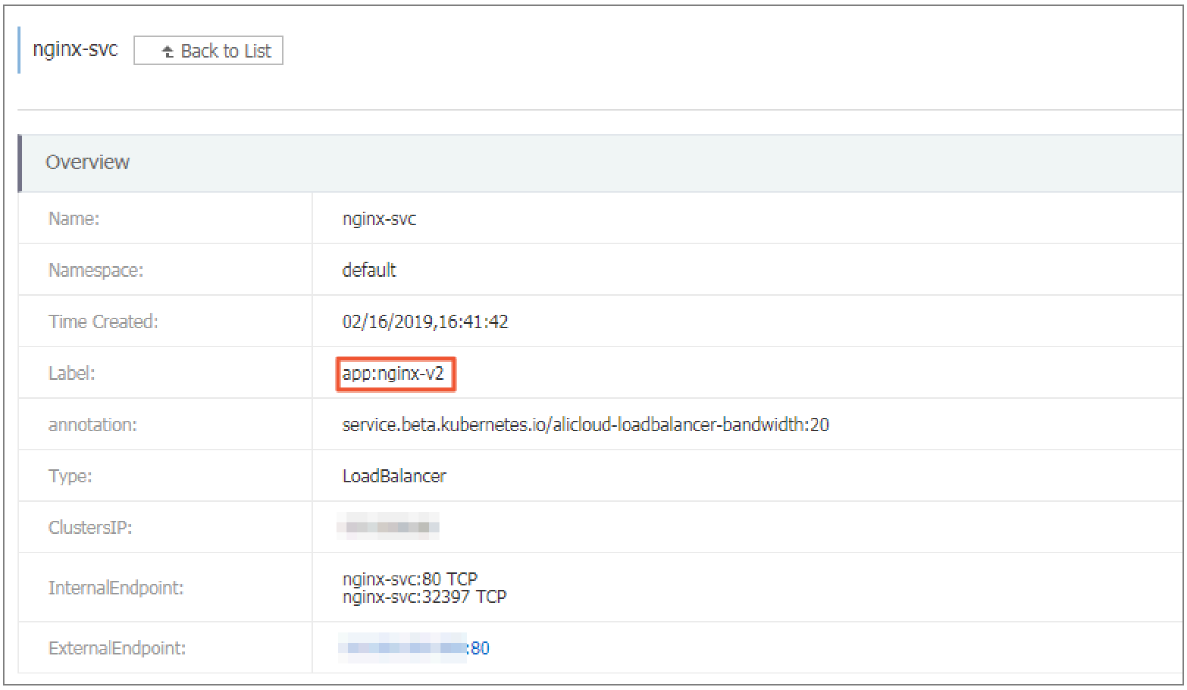
You can update a service in the Container Service console or Kubernetes dashboard.

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. Then, click OK.



5. Select the target service from the service list, and then click Details on the right to view the changes of the service. In this example, the service tag is changed.



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.

1.7.13 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_40*.
- You have created a service successfully. For more information, see *#unique_96*.

Procedure

1. Log on to the *Container Service console*.
2. Click Kubernetes >Application > > Servicein 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.

1.7.14 Use an application trigger

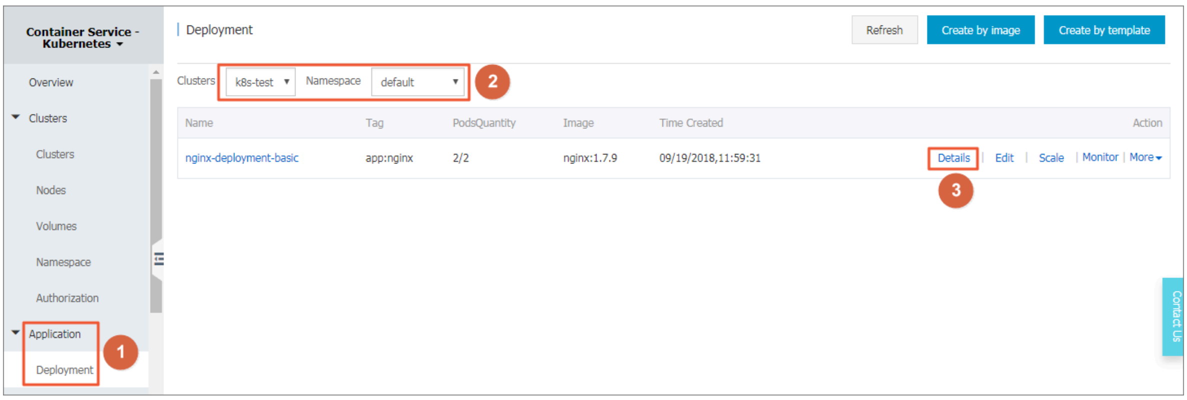
Alibaba Cloud Container Service Kubernetes supports the application trigger function. You can use an application trigger in many ways.

Prerequisites

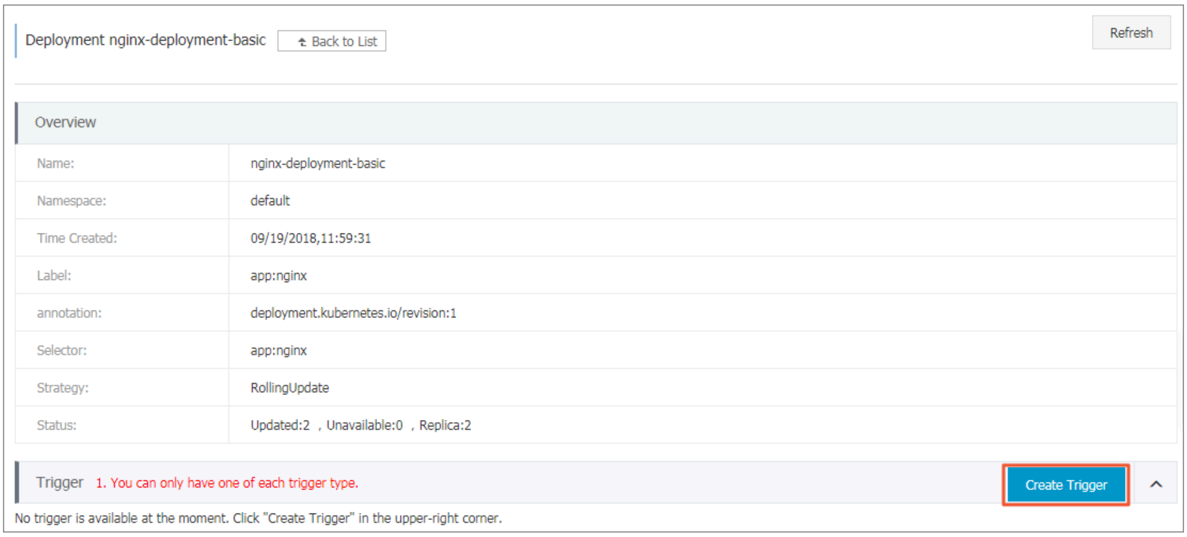
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created an application that is used to create an application trigger and test the trigger. In this example, create an nginx application.

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.

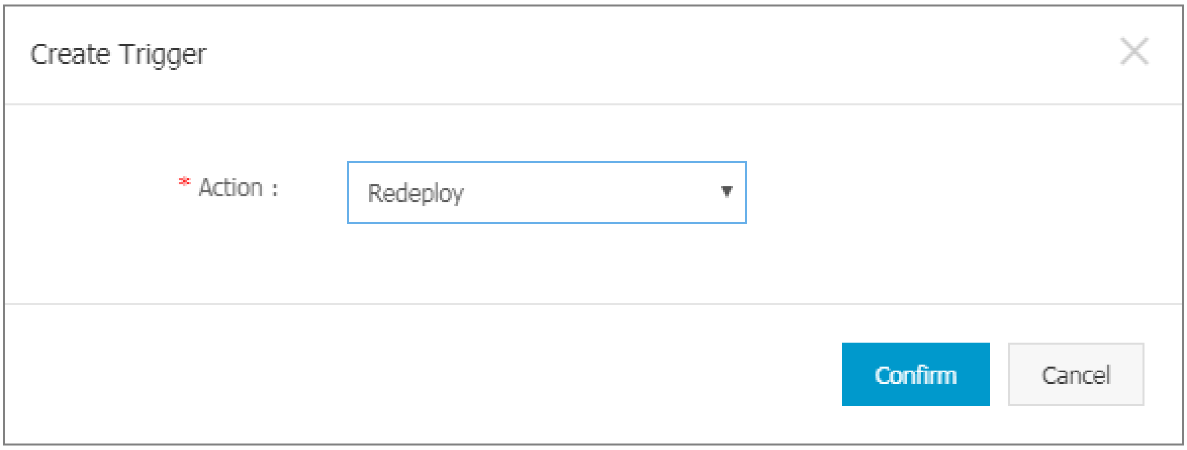


3. On the nginx application details page, click Create Trigger on the right side of the trigger bar.

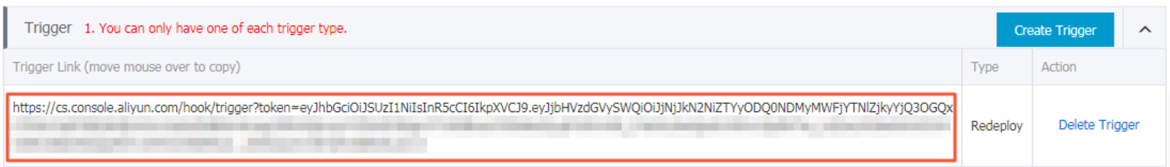


4. In the pop-up dialog box, click Redeploy and click Confirm.

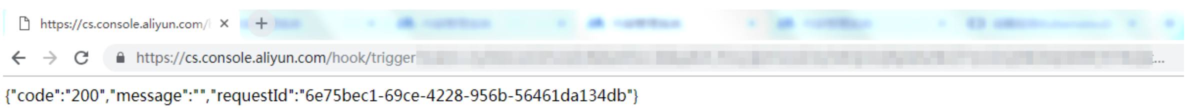
> **Note:**
> Currently, only the redeploy action is supported.



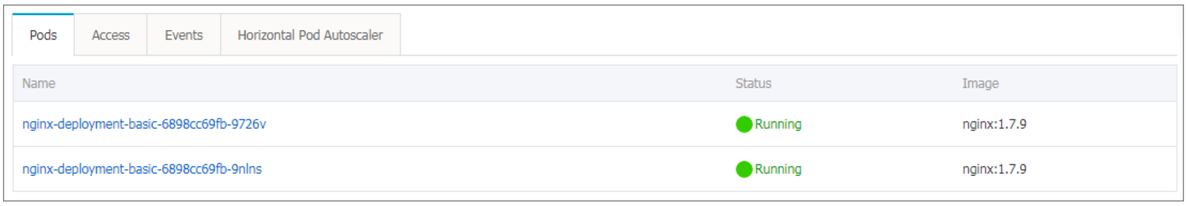
After the trigger is created, a trigger link is displayed in the trigger bar on the nginx application detail page.



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



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



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

What's next

You can call a trigger by using GET or POST in a third-party system. For example, you can run the `curl` command to call a trigger.

Call the redeploy trigger as follows:

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

1.7.15 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.

1.7.16 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.

1.7.17 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_40*.

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.

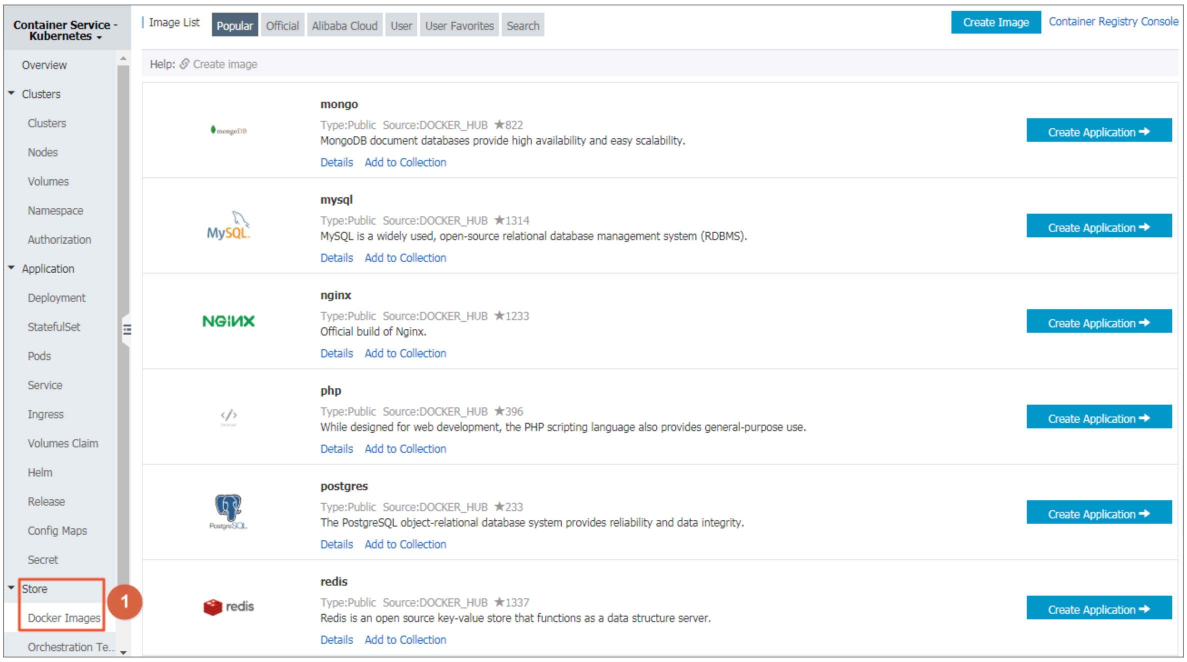
6. Click the pod name to view the pod details.

   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`.

1.7.18 View image list

Procedure

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



You can view the image category.

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

1.7.19 Use an image secret

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

Prerequisites

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

Context

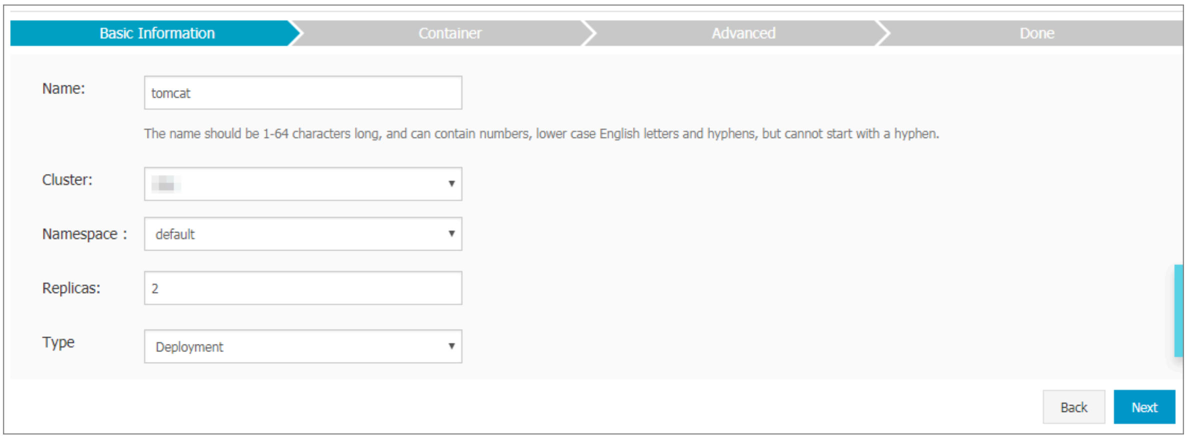
When you use a private image to create an application, you have to configure a secret for the image to secure the image. In the Container Service console, you can deliver the identity authentication information of the private image repository to Kubernetes through a secret of the docker-registry type.

Procedure

1. 登录容器服务管理控制台。
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.

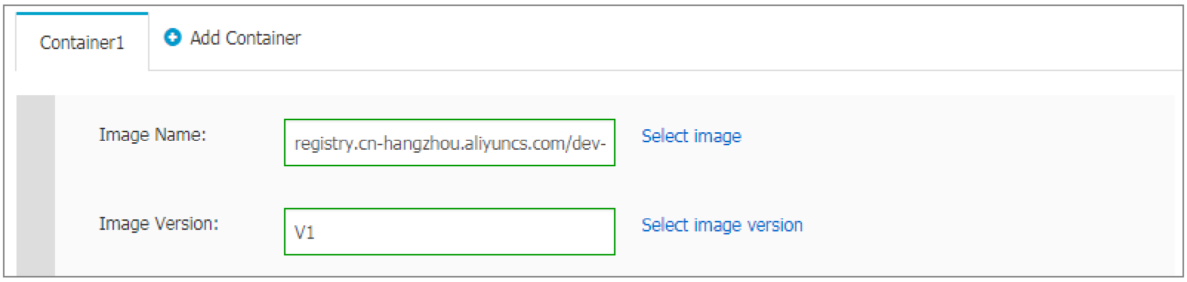
4. Configure containers.

   Note:
   This example describes only the configuration of the container image secret. For more information about container configuration, see *Create a deployment application by using an image*.
5. On the container configuration page, configure the image name first. Enter the private image address in the Image Name box. The format is `domainname/namespace/imagename`.

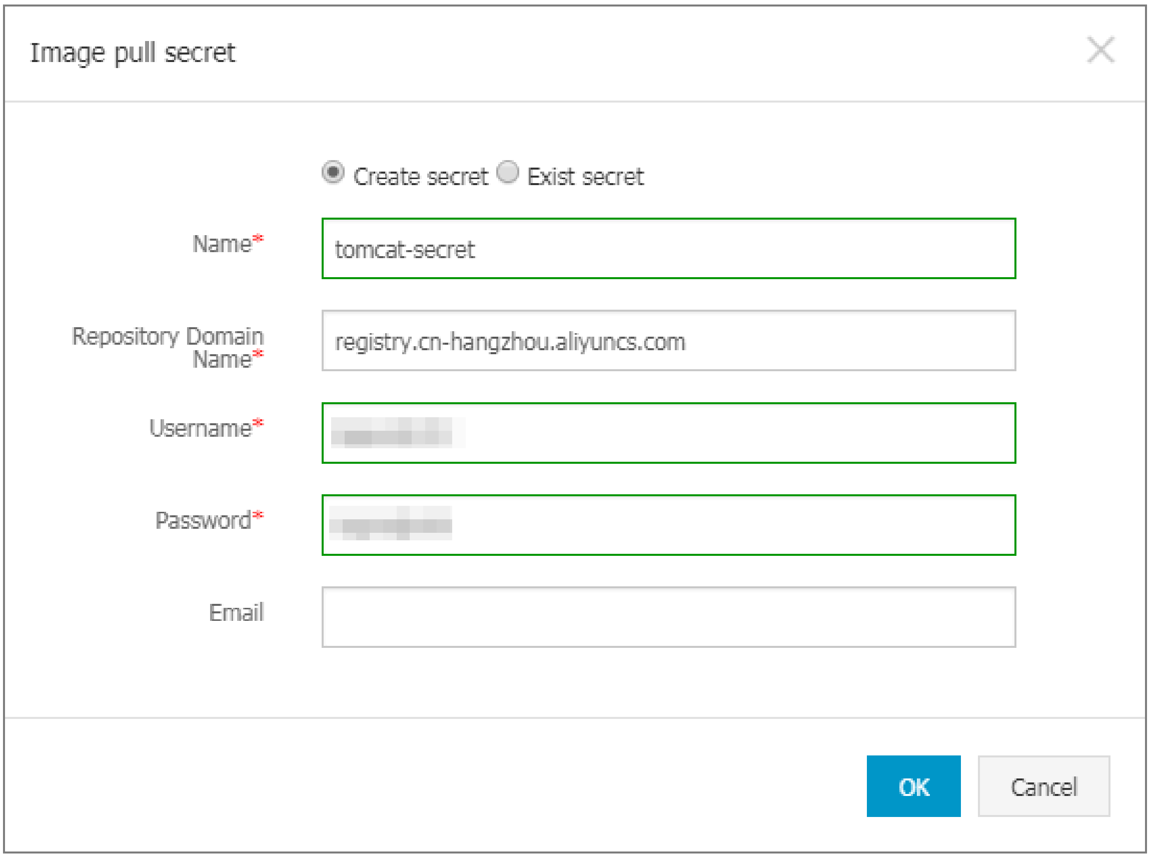
   Note:

Public images do not require image secrets.

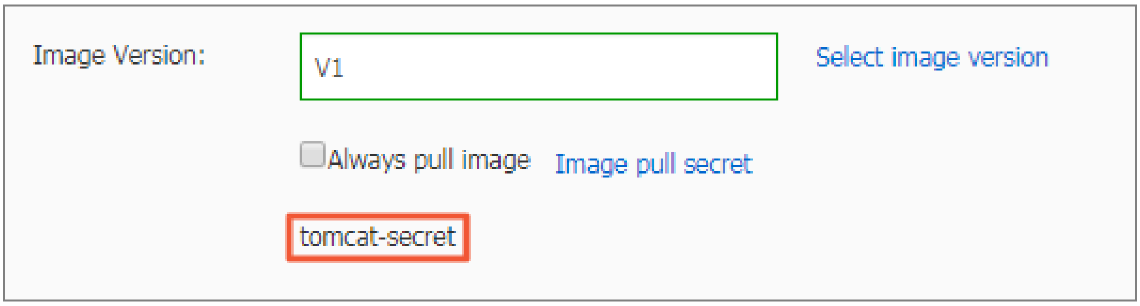
6. In the image version box, enter the private image address version.



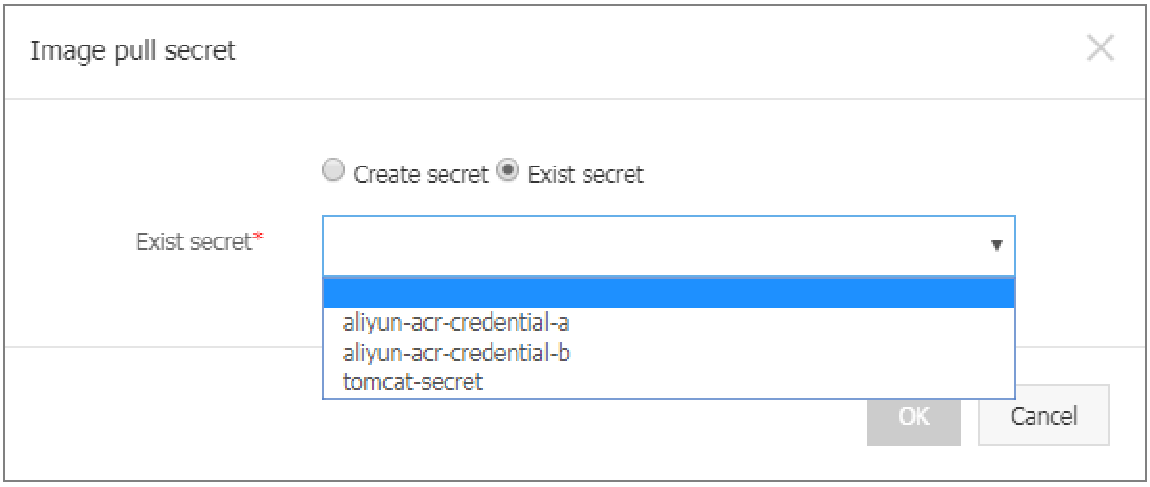
7. Click Image pull secret.
   - Select Create secret.
     - Name: Specifies the secret name. You can define it by yourself.
     - Repository Domain Name: Specified the Docker repository address. If you enter the Alibaba Cloud Container Service image repository in the image name box, the system automatically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



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

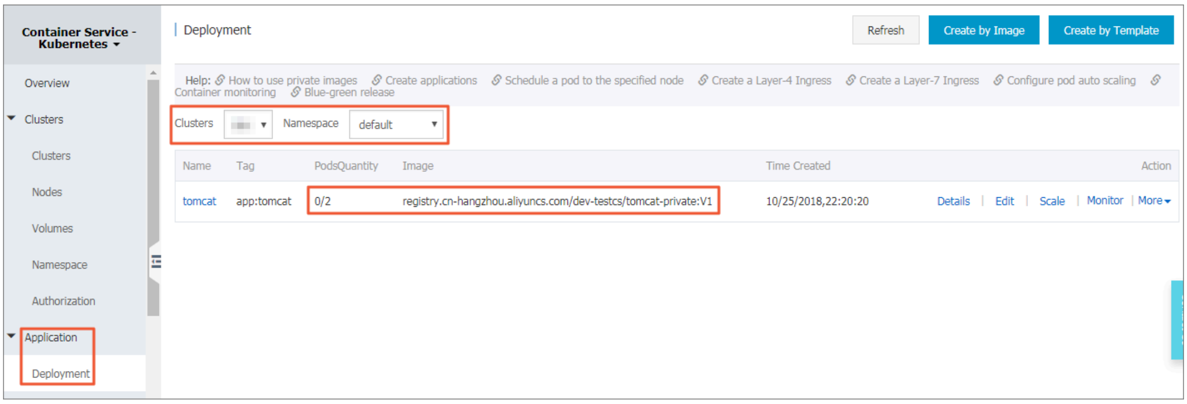


- You can also click Exist secret. You can pre-create a container image secret by using command lines or a YAML file. For information, see *How to use private images in Kubernetes clusters* and *Use a private image repository to create an application*.



8. After you complete the container configuration, click Next.
9. Follow the page guide to complete other configurations, and then click Create.
10. Click 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.



1.8 Network management

1.8.1 Networks supported by Alibaba Cloud Container Service for Kubernetes

This topic describes the networks supported by Alibaba Cloud Container Service for Kubernetes.

Container networks

Container Service provides a stable and high-performance container network through its deep integration of the Kubernetes network and Alibaba Cloud Virtual Private Cloud (VPC). Container Service supports the following types of interconnections:

- Pods within a container cluster can access each other.
- A pod can access a service within a container cluster.
- An Elastic Compute Service (ECS) instance can access a service within a container cluster.
- A pod can directly access an ECS instance (*) in the same VPC.
- An ECS instance can directly access a pod (*) in the same VPC.

Note:
The asterisk (*) indicates that you need to set a valid security group rule.

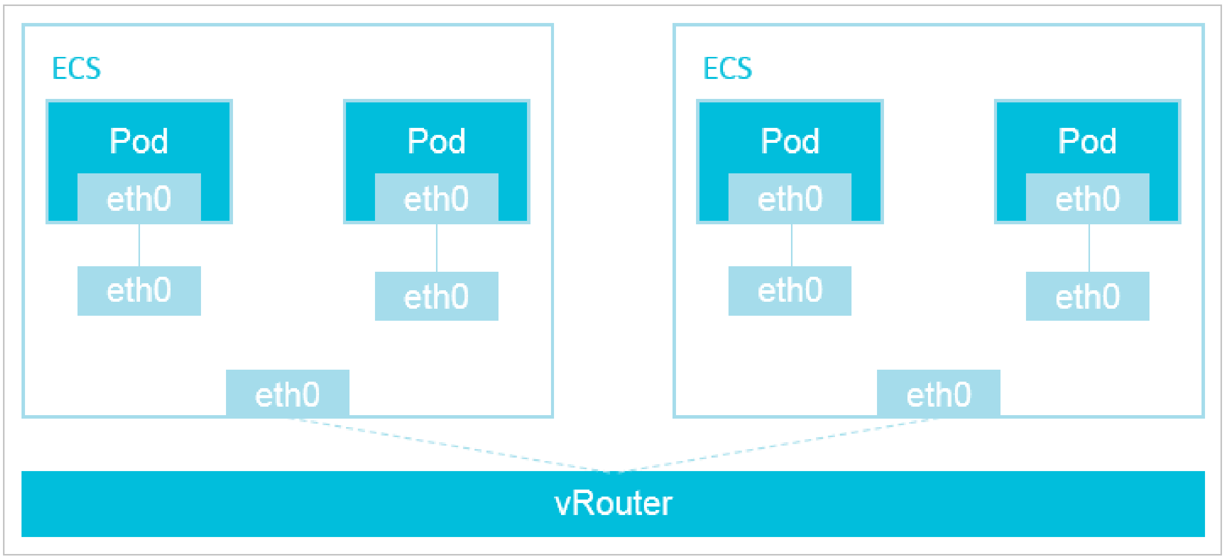
1.8.2 Terway network plugin

This topic describes how to use the Terway network plugin in a Kubernetes cluster that runs on Alibaba Cloud Container Service.

Terway network plugin

Terway, a network plugin developed by Alibaba Cloud Container Service, is fully compatible with Flannel, and provides the following features:

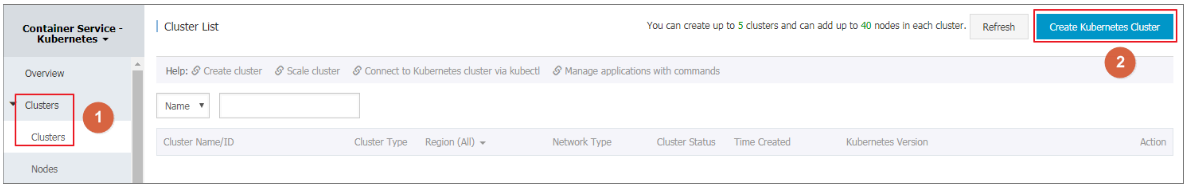
- Allocates Alibaba Cloud Elastic Network Interfaces (ENIs) to containers.
- Defines the access policies for containers according to the Kubernetes Network Policy. This network plugin is also compatible with the Calico Network Policy.



If you install the Terway network plugin in a Kubernetes cluster, each pod then has its own network stack and an IP address. Packets between pods on one ECS instance are forwarded directly by the instance. Packets between pods on different ECS instances are forwarded through the VRouter of a VPC. The Terway network plugin delivers high communication performance because it does not use tunneling technologies such as VXLAN to encapsulate packets.

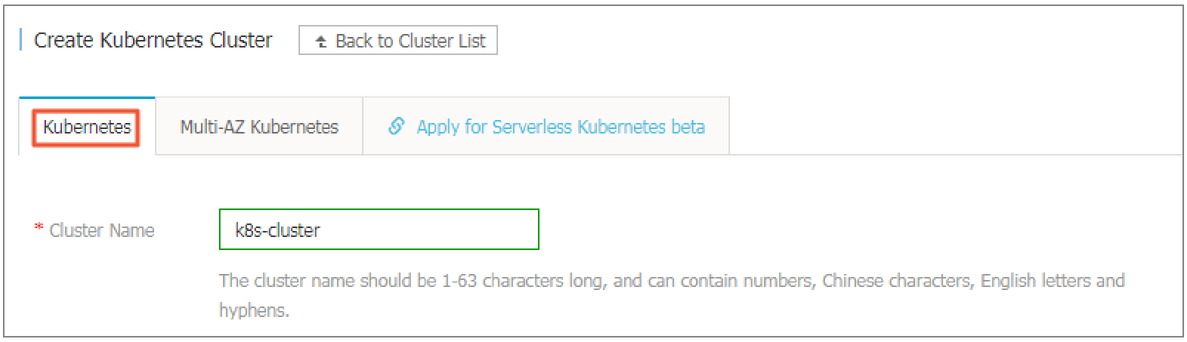
Use the Terway network plugin

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

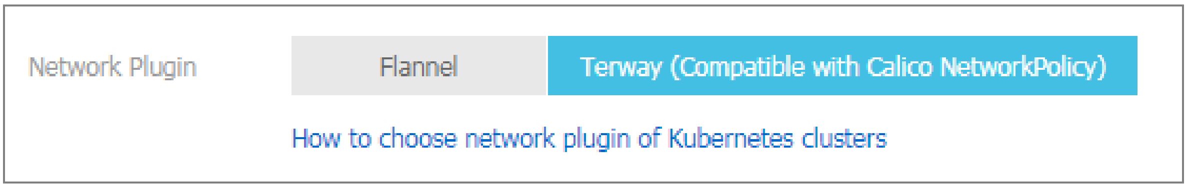


By default, the Create Kubernetes Cluster page is displayed.

> **Note:**
> In this example, a dedicated Kubernetes cluster is created. For more information, see *Create a Kubernetes cluster*.



4. Select the Terway network plugin.



Flannel and Terway

Alibaba Cloud Container Service for Kubernetes provides two types of network plugins for you to create a Kubernetes cluster: Terway and Flannel.

> **Note:**
> For how to select a network plugin, see *Do I select the Terway or Flannel plugin for my Kubernetes cluster network?*

- Flannel: a simple and stable community *Flannel* CNI plugin. Flannel can interoperate with the high-speed network of Alibaba Cloud VPC to provide a high-performance and stable container network for clusters. However, it provides a limited amount of features. For example, it does not support the Kubernetes Network Policy.
- Terway: a network plugin developed by Alibaba Cloud Container service. It is fully compatible with Flannel, and can allocate Alibaba Cloud Elastic Network Interfaces (ENIs) to containers. It can also define the access policies between containers according to the Kubernetes Network Policy. In addition, you can use this network plugin to limit the bandwidth traffic of a single container. If you do not need to use the Network Policy, we recommend that you select Flannel. In other cases, we recommend that you select Terway.

  > **Note:**

- Terway provides the same Network Policy as Calico because Terway is integrated with the Felix component of Calico. If you create a cluster to use Calico, you can use Terway to switch to Alibaba Cloud Container Service for Kubernetes.
- Terway is integrated with the Felix component V2.6.6.

1.8.3 Allocate an ENI to a pod

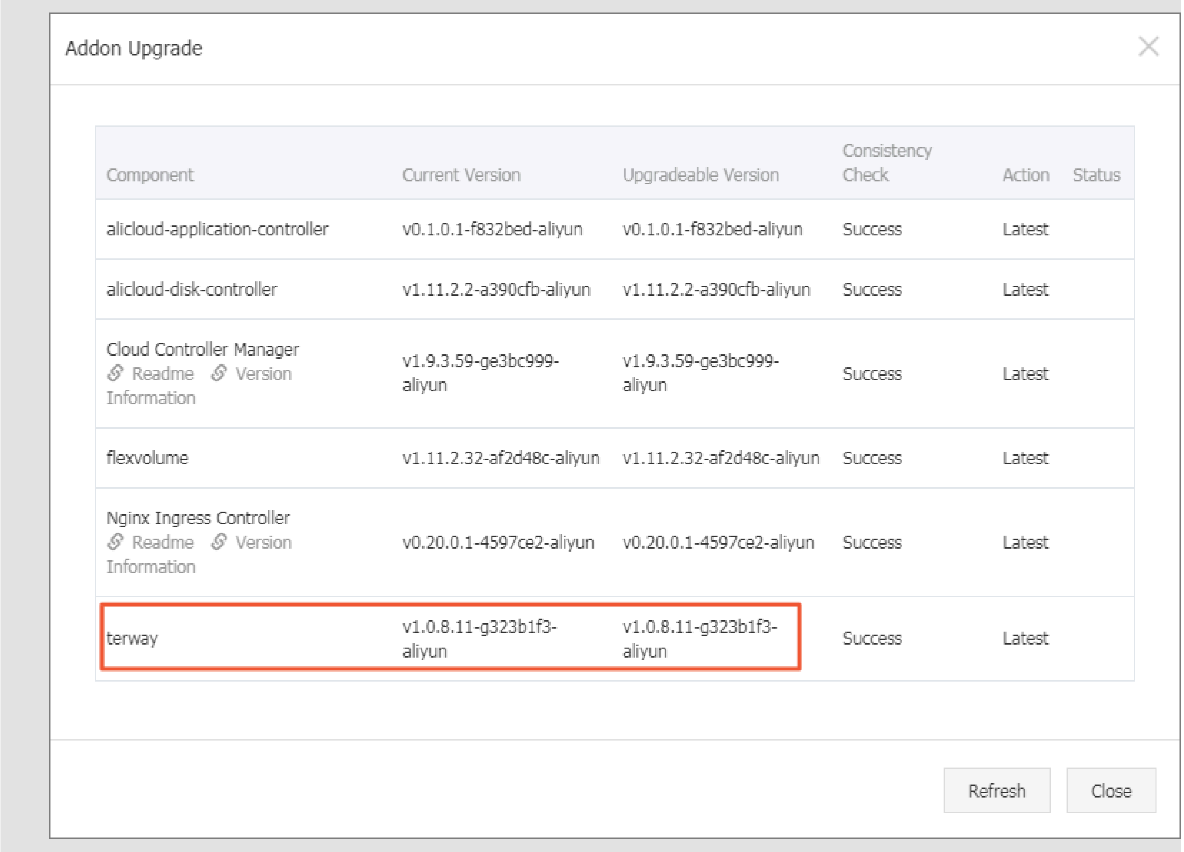
This topic describes how to allocate an Elastic Network Interface (ENI) to a pod.

Context

- When you create a Kubernetes cluster, you need to select Network Plugin as Terway. For more information, see *Create a Kubernetes cluster*.
- If you use a Kubernetes cluster that is installed with the Terway network plugin, you must make sure that the Terway plugin is V1.0.0.1 or later.

Note:

1. Log on to the Container Service console, click Clusters under the Kubernetes menu.
2. In the action column of the target cluster, choose More > Addon Upgrade.
3. On the Addon Upgrade page, view your current version of Terway.
4. Determine whether to upgrade according to Current Version and Upgradeable Version. If you want to upgrade Terway, click Upgrade in the action column.



Procedure

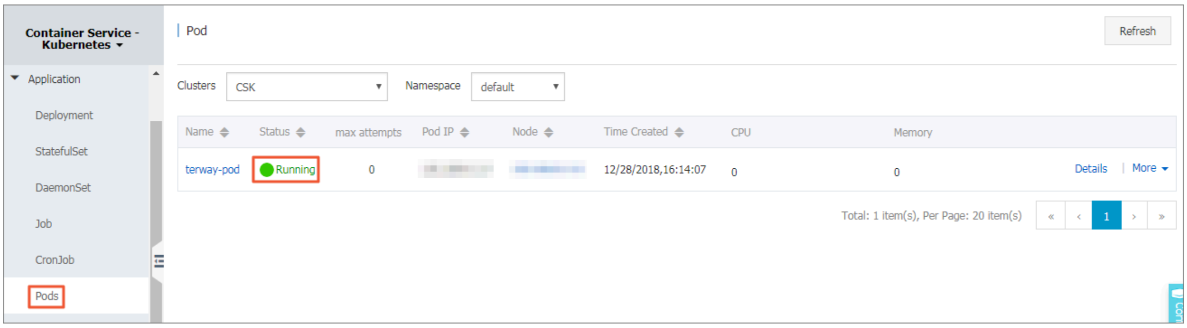
1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose Application > Deployment.
3. In the upper-right corner, click Create by Template.

   You can use the following YAML template to create a pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: terway-pod
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    resources:
      limits:
        aliyun/eni: 1
```

Result

1. In the left-side navigation pane under Kubernetes, choose Application > Pods. The pod named terway-pod is displayed.



2. In the left-side navigation pane under Kubernetes, click Clusters.
3. Click the name of the target cluster to view the cluster details.
4. In the Cluster Resource area, click VPC to view the VPC CIDR block of the cluster.
5. Run the following command to obtain the IP address of the deployed pod and verify that the IP address is within the VPC CIDR block of the cluster:

```
$ kubectl get pod -o wide
```

1.8.4 Use a network policy

Prerequisites

- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have selected the Terway network plugin when creating the Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have connected to the Kubernetes cluster by using kubectl, see *Connect to a Kubernetes cluster by using kubectl*.

Verify that an Nginx service is accessible to pods

1. Run the following command to create an Nginx application and expose it through a service named Nginx:

```
$ kubectl run nginx --image=nginx
deployment.apps/nginx created
$ kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
nginx-64f497f8fd-znbxb   1/1     Running   0          45s

$ kubectl expose deployment nginx --port=80
service/nginx exposed
$ kubectl get service
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   172.19.0.1     <none>        443/TCP    3h
```

```
nginx         ClusterIP   172.19.8.48   <none>        80/TCP    10s
```

2. Run the following command to create a pod named busybox and use the pod to access the Nginx service created in step 1:

```
$ kubectl run busybox --rm -ti --image=busybox /bin/sh
kubectl run --generator=deployment/apps.v1beta1 is DEPRECATED and
will be removed in a future version. Use kubectl create instead.
If you don't see a command prompt, try pressing enter.
/ # wget nginx
Connecting to nginx (172.19.8.48:80)
index.html           100% |
********************************************************************************
   612  0:00:00 ETA
/ #
```

Use a network policy to set the Nginx service to be accessible only to a specifically labeled application

1. Run the following command to create a`policy.yaml` file:

```
$ vim policy.yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: access-nginx
spec:
  podSelector:
    matchLabels:
      run: nginx
  ingress:
  - from:
    - podSelector:
        matchLabels:
          access: "true"
```

2. Run the following command to create a network policy according to the `policy.yaml` file created in step 1:

```
$ kubectl apply -f policy.yaml
networkpolicy.networking.k8s.io/access-nginx created
```

3. Run the following command to verify that the Nginx service cannot be accessed if you do not define any access label in the command:

```
$ kubectl run busybox --rm -ti --image=busybox /bin/sh
If you don't see a command prompt, try pressing enter.
/ # wget nginx
Connecting to nginx (172.19.8.48:80)
wget: can't connect to remote host (172.19.8.48): Connection timed
out
```

```
/ #
```

4. Run the following command to verify that the Nginx service can be accessed if an access label is defined in the command:

```
$ kubectl run busybox --rm -ti --labels="access=true" --image=
busybox /bin/sh
If you don't see a command prompt, try pressing enter.
/ # wget nginx
Connecting to nginx (172.19.8.48:80)
index.html           100% |
*****************************************************************************
   612  0:00:00 ETA
/ #
```

Use a network policy to specify a source IP CIDR block that can access a service exposed by an SLB service over the Internet

1. Run the following command to create an Alibaba Cloud SLB service for the preceding Nginx application, that is, specify `type=LoadBalancer` to expose the Nginx service to the Internet:

```
$ vim nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: nginx-slb
spec:
  externalTrafficPolicy: Local
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer

$ kubectl apply -f nginx-service.yaml
service/nginx-slb created

$ kubectl get service nginx-slb
NAME        TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)
       AGE
nginx-slb   LoadBalancer   172.19.12.254   47.110.200.119   80:32240
/TCP   8m
```

2. Run the following command to verify that the IP address of the created SLB service, that is, 47.110.200.119, cannot be accessed:

```
$ wget 47.110.200.119
--2018-11-21 11:46:05--  http://47.110.200.119/
```

```
Connecting to 47.110.200.119:80... failed: Connection refused.
```

Note:

Access failure occurs due to the following reasons:

- You have configured access to the Nginx service only for the applications labeled with `access=true`.
- You have attempted to access the IP address of the SLB instance from outside the Kubernetes system. This is different from *Use a network policy to set the Nginx service to be accessible only to a specifically labeled application*.

Solution: Modify the network policy and add a source IP CIDR block that is allowed to access the Nginx service.

3. Run the following command to view your local IP address:

```
$ curl myip.ipip.net

IP address: 10.0.0.1 from: China Beijing Beijing        #The local
IP address varies by devices.
```

4. Run the following command to modify the created *policy.yaml* file:

```
$ vim policy.yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: access-nginx
spec:
  podSelector:
    matchLabels:
      run: nginx
  ingress:
  - from:
    - podSelector:
        matchLabels:
          access: "true"
    - ipBlock:
        cidr: 100.64.0.0/10
    - ipBlock:
        cidr: 10.0.0.1/24      #Set the CIDR block to which the
 local IP address belongs. This is an example. Set the required
parameters according to your device.

$ kubectl apply -f policy.yaml
networkpolicy.networking.k8s.io/access-nginx unchanged
```

Note:

- The outgoing interface of a network may have multiple IP addresses. We recommend that you specify an entire CIDR block.

- The SLB health check address belongs to the `100.64.0.0/10` CIDR block. Therefore, you must specify the `100.64.0.0/10` CIDR block.

5. Run the following command to verify that the Nginx service can be accessed:

```
$  kubectl run busybox --rm -ti --labels="access=true" --image=
busybox /bin/sh

If you don't see a command prompt, try pressing enter.
/ # wget 47.110.200.119
Connecting to 47.110.200.119 (47.110.200.119:80)
index.html           100% |
***********************************************************|   612
0:00:00 ETA
/ #
```

Use a network policy to set a pod that can access only *www.aliyun.com*

1. Run the following command to obtain the IP address list resolved from the domain name of *www.aliyun.com*:

```
$ dig +short www.aliyun.com
www-jp-de-intl-adns.aliyun.com.
www-jp-de-intl-adns.aliyun.com.gds.alibabadns.com.
v6wagbridge.aliyun.com.
v6wagbridge.aliyun.com.gds.alibabadns.com.
106.11.93.21
140.205.32.4
140.205.230.13
140.205.34.3
```

2. Run the following command to create a*busybox-policy* file:

```
$ vim busybox-policy.yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: busybox-policy
spec:
  podSelector:
    matchLabels:
      run: busybox
  egress:
  - to:
    - ipBlock:
        cidr: 106.11.93.21/32
    - ipBlock:
        cidr: 140.205.32.4/32
    - ipBlock:
        cidr: 140.205.230.13/32
    - ipBlock:
        cidr: 140.205.34.3/32
  - to:
    - ipBlock:
        cidr: 0.0.0.0/0
    ports:
    - protocol: UDP
```

```
      port: 53
```

Note:

In the preceding *busybox-policy* file, an egress rule is set to specify the CIDR blocks that can be accessed by cluster applications. You need to set the condition that UDP requests are allowed. Otherwise, DNS resolution will fail.

3. Run the following command to create a network policy according to the *busybox-policy* file:

```
$ kubectl apply -f busybox-policy.yaml
networkpolicy.networking.k8s.io/busybox-policy created
```

4. Run the following command to verify that no website (for example, *www.google.com*) can be accessed except for *www.aliyun.com*:

```
$ kubectl run busybox --rm -ti --image=busybox /bin/sh
If you don't see a command prompt, try pressing enter.
/ # wget www.google.com
Connecting to www.google.com (64.13.192.74:80)
wget: can't connect to remote host (64.13.192.74): Connection timed
out
```

5. Run the following command to verify that *www.aliyun.com* can be accessed:

```
/ # wget www.aliyun.com
Connecting to www.aliyun.com (140.205.34.3:80)
Connecting to www.aliyun.com (140.205.34.3:443)
wget: note: TLS certificate validation not implemented
index.html           100% |
************************************************************|  462k
0:00:00 ETA
/ #
```

1.9 Server Load Balancer and Ingress management

1.9.1 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.

1.9.2 Access services by using Server Load Balancer

This topic describes how to access services by using Alibaba Cloud Server Load Balancer (SLB).

Check the cloud-controller-manager version

If you specify an existing SLB in a cluster that has a cloud-controller-manager component of v1.9.3 or later versions, the system does not process listeners for this SLB by default. You must manually configure listeners for this SLB.

To view the cloud-controller-manager version, run the following command:

```
root@master # kubectl get po -n kube-system -o yaml|grep image:|grep
cloud-con|uniq

  image: registry-vpc.cn-hangzhou.aliyuncs.com/acs/cloud-controller-
manager-amd64:v1.9.3
```

Use a command-line tool

Method 1

1. Create an Nginx application by using a command-line tool.

```
root@master # kubectl run nginx --image=registry.aliyuncs.com/acs/
netdia:latest
root@master # kubectl get po
NAME                                   READY     STATUS    RESTARTS
  AGE
nginx-2721357637-dvwq3                 1/1       Running   1
  6s
```

2. Create an SLB service for the Nginx application and specify `type=LoadBalancer` to expose the Nginx service to the Internet.

```
root@master # kubectl expose deployment nginx --port=80 --target-
port=80 --type=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 a browser to access your Nginx service.

Method 2

1. Save the following yml code to the `nginx-svc.yml` file:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nignx
  name: nginx-01
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

2. Run the `kubectl apply -f nginx-svc.yml` command.

```
root@master # kubectl apply -f nginx-svc.yml
root@master # kubectl get service
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(
S)        AGE9d
ngi-01nx        LoadBalancer   172.19.9.243    101.37.192.129   80:
32325/TCP   3h
```

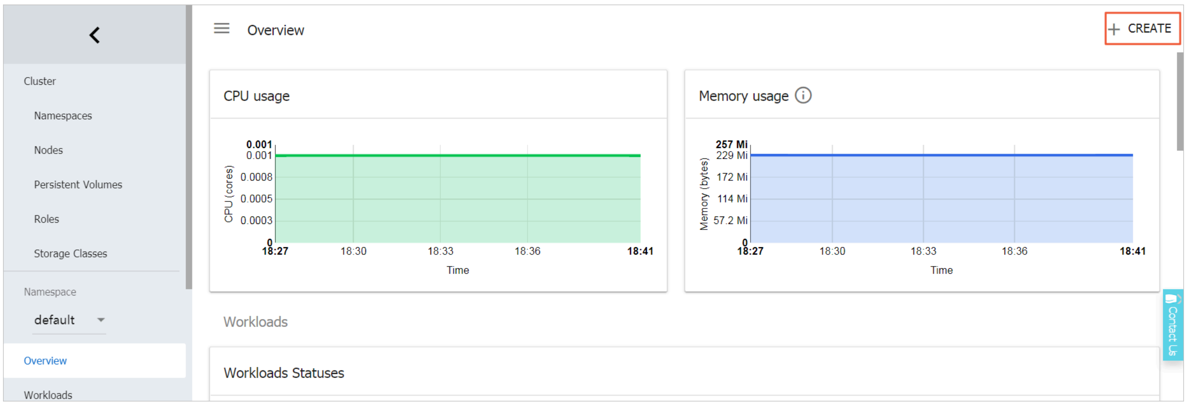
3. Visit `http://101.37.192.129` in a browser to access your Nginx service.

Use the Kubernetes dashboard

1. Save the following yml code to the `nginx-svc.yml` file:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: http-svc
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

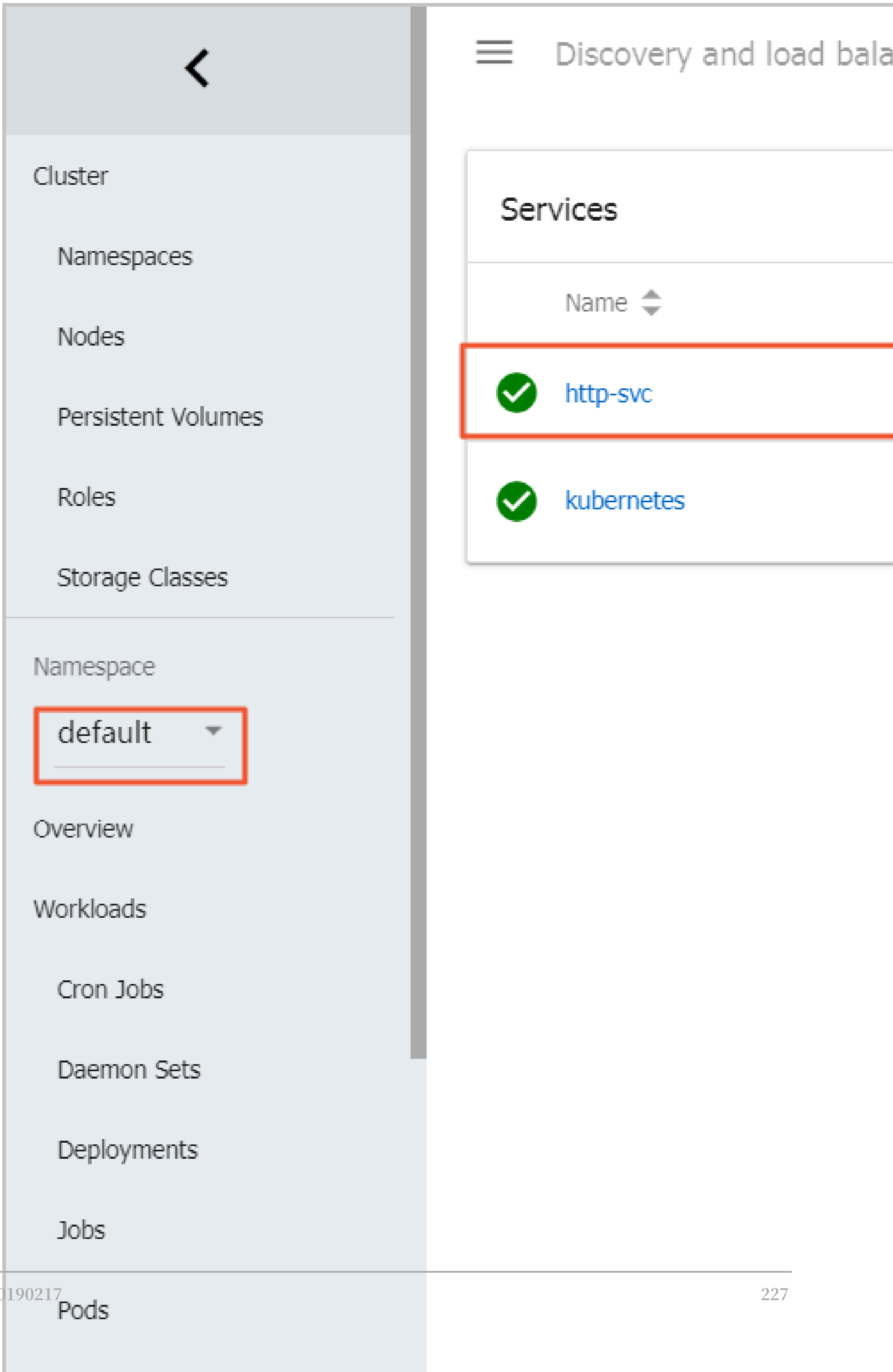
2. Log on to the *Container Service console* and click Dashboard on the right of the target cluster.
3. Click CREATE in the upper-right corner to create an application.



4. Click the CREATE FROM FILE tab. Select the *nginx-svc.yml* file you saved.
5. Click UPLOAD.

   An SLB instance that points to the created Nginx application is created. The service name is `http-svc`.
6. In the left-side navigation pane on the dashboard page, select the default namespace, and then click `Services`.

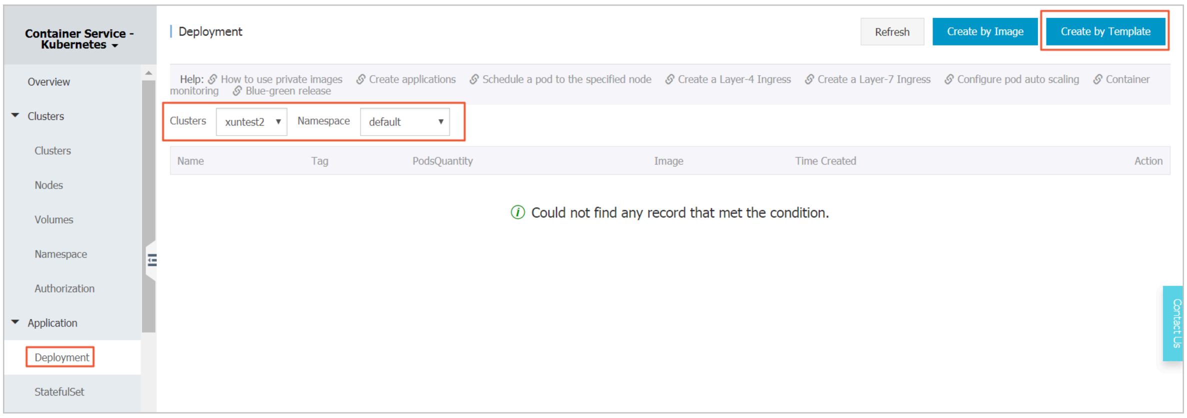
   You can view the created Nginx service `http-svc` and the SLB address `http://114.55.79.24:80`.


Discovery and load bala
Cluster
Namespaces
Nodes
Persistent Volumes
Roles
Storage Classes
Namespace
default
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Services
Name
http-svc
kubernetes

7. Open this address in your browser to access the service.

Use the Container Service console

1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose Application > Deployment.
3. Select the target cluster and namespace, and then click Create by Template in the upper-right corner.

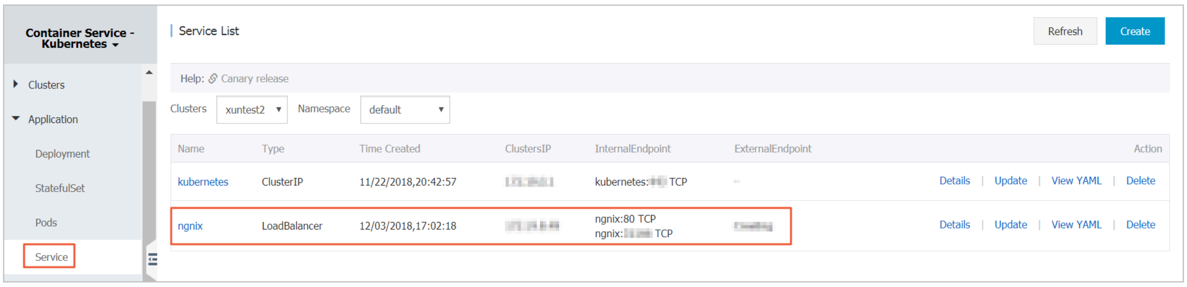


4. Select the custom Resource Type and then copy the following code to the Template.

```
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: ngnix
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

5. Click DEPLOY.
6. Click Kubernetes Dashboard to check the deployment progress on the dashboard page.

Alternatively, choose Application > Service in the left-side navigation pane, and select the target cluster and namespace to view the deployed service.



More information

Alibaba Cloud SLB also supports a lot of parameters such as health checks, billing methods, and SLB types. For more information, see *SLB configuration parameters*.

Annotations

Alibaba Cloud supports plenty of SLB features by using `annotations`.

Use an existing intranet SLB instance

You must specify three annotations. Replace "your-loadbalancer-id" with your SLB instance ID.

Note:

Multiple Kubernetes services can reuse the same SLB instance.

- The SLB instances created by Kubernetes through a service cannot be reused. Otherwise, the reused SLB instances may be removed incidentally. Only the SLB instances manually created in the console or created by calling API can be reused.
- The multiple services that reuse the same SLB instance cannot have the same frontend listening port. Otherwise, port conflicts will occur.
- If you reuse an SLB instance, you must use the listener name and the virtual server group name to martk the SLB instance. Do not modify the listener names or virtual server group names.
- You can modify SLB instance names.
- SLB instances cannot be reused across clusters.

```
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"
    service.beta.kubernetes.io/alicloud-loadbalancer-force-override-
listeners: "true"
  labels:
    run: nginx
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  sessionAffinity: None
  type: LoadBalancer
```

Create an HTTP-type SLB instance

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port: "
http:80"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

Create an HTTPS-type SLB instance

You must first create a certificate in the Alibaba Cloud console before creating an HTTPS-type SLB instance by using the following template (the certificate ID is required by the annotations in the template):

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-cert-id: "your-
cert-id"
    service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port: "
https:443"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
```

```
  selector:
    run: nginx
  sessionAffinity: None
  type: LoadBalancer
```

Limit SLB instance bandwidth

Only the total bandwidth of the SLB instance can be limited, and all listeners share this bandwidth. For more information, see 共享实例带宽.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-charge-type: "
paybybandwidth"
    service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth: "100"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

Specify the SLB instance specification

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-spec: "slb.s1.
small"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

Use an existing SLB instance

By default, listeners are not overridden if you use an existing SLB instance. To forcibly override the existing listeners, set `service.beta.kubernetes.io/alicloud-loadbalancer-force-override-listeners` to `true`.

Note:

Multiple Kubernetes services can reuse the same SLB instance.

- The SLB instances created by Kubernetes through a service cannot be reused. Otherwise, the reused SLB instances may be removed incidentally. Only the SLB instances manually created in the console or created by calling API can be reused.
- The multiple services that reuse the same SLB instance cannot have the same frontend listening port. Otherwise, port conflicts will occur.
- If you reuse an SLB instance, you must use the listener name and the virtual server group name to martk the SLB instance. Do not modify the listener names or virtual server group names.
- You can modify SLB instance names.
- SLB instances cannot be reused across clusters.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-id: "your_loadb
alancer_id"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer: LoadBalancer
```

Use an existing SLB instance and forcibly override existing listeners

If you forcibly override the existing listeners, they are removed.

Note:

Multiple Kubernetes services can reuse the same SLB instance.

- The SLB instances created by Kubernetes through a service cannot be reused. Otherwise, the reused SLB instances may be removed incidentally. Only the SLB instances manually created in the console or created by calling API can be reused.
- The multiple services that reuse the same SLB instance cannot have the same frontend listening port. Otherwise, port conflicts will occur.
- If you reuse an SLB instance, you must use the listener name and the virtual server group name to martk the SLB instance. Do not modify the listener names or virtual server group names.
- You can modify SLB instance names.

- **SLB instances cannot be reused across clusters.**

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-id: "your_loadb
alancer_id"
    service.beta.kubernetes.io/alicloud-loadbalancer-force-override-
listeners: "true"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancere: LoadBalancer
```

Use the Worker node with specified labels as a backend server

Use a comma (,) to separate two labels, for example, K1: V1, K2: V2.

The relationship between multiple labels is `and`.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-backend-label: "
failure-domain.beta.kubernetes.io/zone:ap-southeast-5a"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

Set the session persistence timeout for a TCP-type SLB instance

The parameter `service.beta.kubernetes.io/alicloud-loadbalancer-persistence-tim` applies only to TCP listeners.

If the SLB instance is configured with multiple TCP listener ports, this parameter setting applies to all the ports by default.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
```

```
    service.beta.kubernetes.io/alicloud-loadbalancer-persistence-
timeout: "1800"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  type: LoadBalancer
```

Set session persistence for HTTP-type and HTTPS-type SLB instances (insert cookie)

Only HTTP-type and HTTPS-type SLB instances support this setting.

If an instance is configured with multiple HTTP or HTTPS listener ports, the session persistence setting applies to all the HTTP or HTTPS listener ports by default.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session: "
on"
    service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session-
type: "insert"
    service.beta.kubernetes.io/alicloud-loadbalancer-cookie-timeout: "
1800"
    service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port: "
http:80"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

Set session persistence for HTTP-type and HTTPS-type SLB instances (server cookie)

Only HTTP-type and HTTPS-type SLB instances support this setting.

If an instance is configured with multiple HTTP or HTTPS listener ports, the session persistence setting applies to all the HTTP or HTTPS listener ports by default.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session: "
on"
    service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session-
type: "server"
```

```
    service.beta.kubernetes.io/alicloud-loadbalancer-cooyour_cookie: "
your_cookie"
    service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port: "
http:80"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

Specify the primary and secondary zones when creating an SLB instance

Support for the primary and secondary zones varies according to region, for example , the ap-southeast-5.

Once created, the primary and secondary zones cannot be changed.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-master-zoneid: "
ap-southeast-5a"
    service.beta.kubernetes.io/alicloud-loadbalancer-slave-zoneid: "ap
-southeast-5a"
  name: nginx
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: LoadBalancer
```

Use the node where the pod is located as a backend server

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
spec:
  externalTrafficPolicy: Local
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
```

```
  type: LoadBalancer
```

Note:

The annotations are case sensitive.

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port`	Use a comma (,) to separate two values, for example, https:443,http:80.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-address-type`	Valid values: `internet` or `intranet`.	`internet`
`service.beta.kubernetes.io/alicloud-loadbalancer-slb-network-type`	SLB instance network type. Valid values: `classic` or `vpc`.	`classic`
`service.beta.kubernetes.io/alicloud-loadbalancer-charge-type`	Valid values: `paybytraffic` or `paybybandwidth`.	`paybytraffic`
`service.beta.kubernetes.io/alicloud-loadbalancer-id`	SLB instance ID. You can specify an existing SLB instance by using `service.beta.kubernetes.io/alicloud-loadbalancer-id`, and existing listeners will be overridden. Note that the SLB instance will not be deleted if you delete the service.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-backend-label`	Use labels to specify the Worker nodes to be mounted to the backend of the SLB instance.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-spec`	SLB instance specification. For more information, see *#unique_115*.	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-persistence-timeout`	Session persistence timeout (in seconds). This parameter setting applies only to TCP listeners and the value can be `0 to 3600`. The default value is `0`, indicating that the session remains disabled. For more information, see *#unique_116*.	`0`
`service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session`	Whether to enable session persistence. Valid value: `on | off`. Note: It applies only to HTTP and HTTPS listeners. For more information, see *#unique_117* and *#unique_118*.	`off`

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session-type`	Method used to handle the cookie. Valid values: · `insert`: Insert the cookie. · `server`: Rewrite the cookie. Note: · It applies only to HTTP and HTTPS listeners. · If you set the value of the parameter `service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session` to `on`, you must specify this parameter. For more information, see *#unique_117* and *#unique_118*.	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-cookie-timeout`	Cookie timeout period (in seconds). Value range: `1 to 86400`. **Note:** If the `service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session` parameter is set to `on` and the `service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session-type` parameter is set to `insert`, this parameter is mandatory. For more information, see *#unique_117* and *#unique_118*.	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-cookie`	Cookie configured on the server. The cookie must be a string of 1 to 200 characters and can only contain ASCII letters and numeric characters. It cannot contain commas (,), semicolons (;), or spaces, and it cannot start with a dollar sign ($). **Note:** If the `service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session` parameter is set to `on` and the `service.beta.kubernetes.io/alicloud-loadbalancer-sticky-session-type` parameter is set to `server`, this parameter is mandatory. For more information, see *#unique_117* and *#unique_118*.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-master-zoneid`	Zone ID of the primary backend server.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-slave-zoneid`	Zone ID of the secondary backend server.	None

Annotation	Description	Default value
`externalTrafficPolicy`	Nodes that can be used as backend servers. Valid values: · `Cluster`: Use all backend nodes as back-end servers. · `Local`: Use the nodes where pods are located as backend servers.	`Cluster`
`service.beta.kubernetes.io/alicloud-loadbalancer-force-override-listeners`	Determines whether to override the listeners when you specify an existing SLB instance.	`false`: Do not override.
`service.beta.kubernetes.io/alicloud-loadbalancer-region`	Region where the SLB instance is located.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth`	SLB instance bandwidth.	`50`
`service.beta.kubernetes.io/alicloud-loadbalancer-cert-id`	ID of a certificate on Alibaba Cloud. You must upload a certificate first.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-flag`	Valid values: `on` | `off`.	The default value is `off`. Modifying this parameter is not required for TCP, because the health check function is enabled for TCP by default and this parameter cannot be set.
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-type`	Health check type. Valid values: `tcp` | `http`. For more information, see *#unique_116*.	`tcp`

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-uri`	URI used for health checks. **Note:** If the health check type is TCP, you do not need to set this parameter. For more information, see *#unique_116*.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-port`	Port used for health checks. Valid values: · `-520`: The backend port configured for the listener is used by default. · `1-65535`: The port opened on the backend server for health checks is used. For more information, see *#unique_116*.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-healthy-threshold`	For more information, see *#unique_116*.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-unhealthy-threshold`	The number of consecutive health check successes before the backend server is determined healthy (from failure to success). Value range: `2 to 10` For more information, see *#unique_116*.	None
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval`	Time interval between two consecutive health checks (seconds). Value range: `1 to 50`. For more information, see *#unique_116*.	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout`	Time period required by waiting for a health check response (in seconds). If the backend ECS instance does not send a valid response within a specified period of time, the system determines that the health check has failed. Value range: `1 to 300`. **Note:** If the value of the parameter `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout` is less than the value of the parameter `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval`, `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout` is invalid and the timeout period equals the value of `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval`. For more information, see *#unique_116*.	None

Annotation	Description	Default value
`service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout`	Time period required by waiting for a health check response (in seconds). If the backend ECS instance does not send a valid response within a specified period of time, the system determines that the health check has failed. Value range: `1 to 300`. Note: If the value of the parameter `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout` is less than that of the parameter `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval`, `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout` is invalid, and the timeout period equals the value of the parameter `service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval`. For more information, see *#unique_117*.	None

1.9.3 Support for Ingress

In Kubernetes clusters, Ingress is a collection of rules that authorize inbound connection to the cluster services and provides you with Layer-7 Server Load

Balancer capabilities. You can provide the Ingress configuration with externally accessible URL, Server Load Balancer, SSL, and name-based virtual host.

Prerequisites

To test the complex routing service, create an Nginx application in this example. You must create the Nginx deployment and multiple services in advance to observe the routing effect. Replace with your own service in the actual test.In the actual test enter your own service.

```
root@master # kubectl run nginx --image=registry.cn-hangzhou.aliyuncs.
com/acs/netdia:latest

root@master # kubectl expose deploy nginx --name=http-svc --port=80 --
target-port=80
root@master # kubectl expose deploy nginx --name=http-svc1 --port=80
 --target-port=80
root@master # kubectl expose deploy nginx --name=http-svc2 --port=80
 --target-port=80
root@master # kubectl expose deploy nginx --name=http-svc3 --port=80
 --target-port=80
```

Simple routing service

Create a simple Ingress service by using the following commands. All the accesses to the */svc* path are routed to the Nginx service. `nginx.ingress.kubernetes.io/rewrite-target: /` redirects the path */svc*to the path */* that can be recognized by backend services.

```
root@master # cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple
  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. 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`.

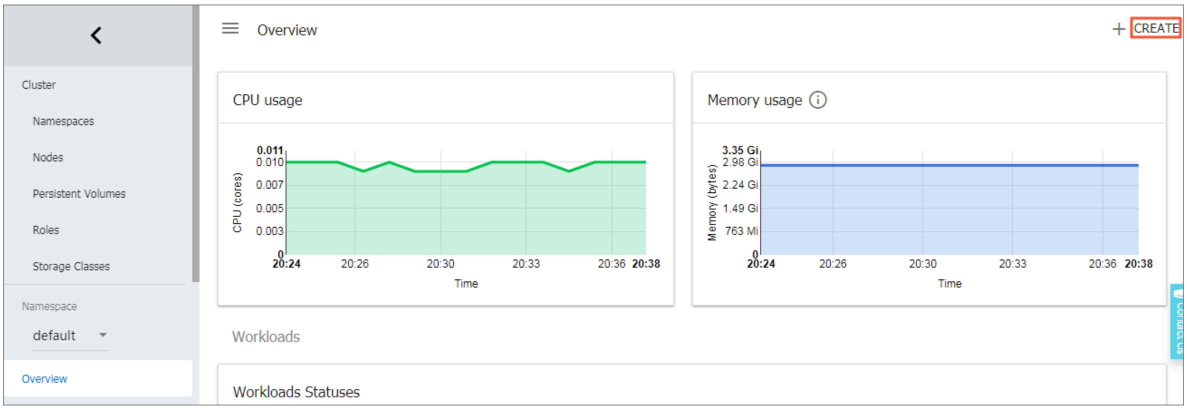
You can also access the HTTPS service by using HTTP. By default, Ingress redirects HTTP access configured with HTTPS to the HTTPS address. Therefore, access to `http://foo.bar.com/foo` will be automatically redirected to `https://foo.bar.com/foo`.

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
```

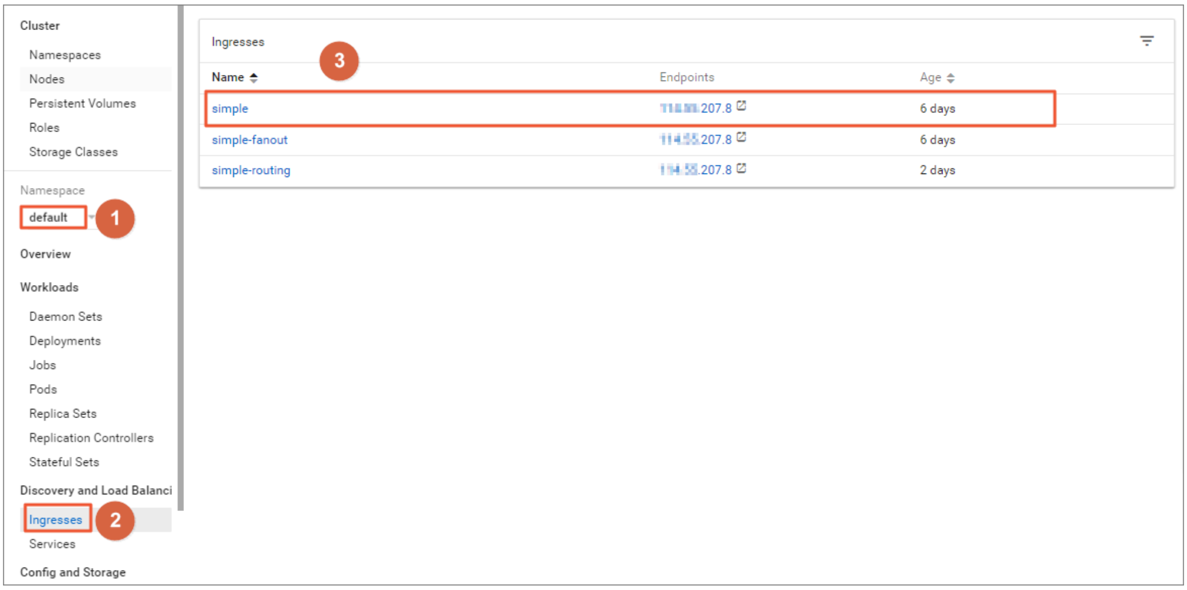
2. Log on to the *Container Service console*. In the left-side navigation pane under Kubernetes, click Clusters. Then click Dashboardon the right of the target cluster.
3. Click CREATE in the upper-right corner to create an application.



4. Click the CREATE FROM FILE tab. Select the `nginx-ingress.yml` file you saved.
5. Click UPLOAD.

   Then an Ingress Layer-7 proxy route will be created to the `http-svc` service.
6. Click default under Namespace in the left-side navigation pane. Click Ingresses in the left-side navigation pane.

   You can view the created Ingress resource and its access address `http://118.178.174.161/svc`.



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

1.9.4 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.

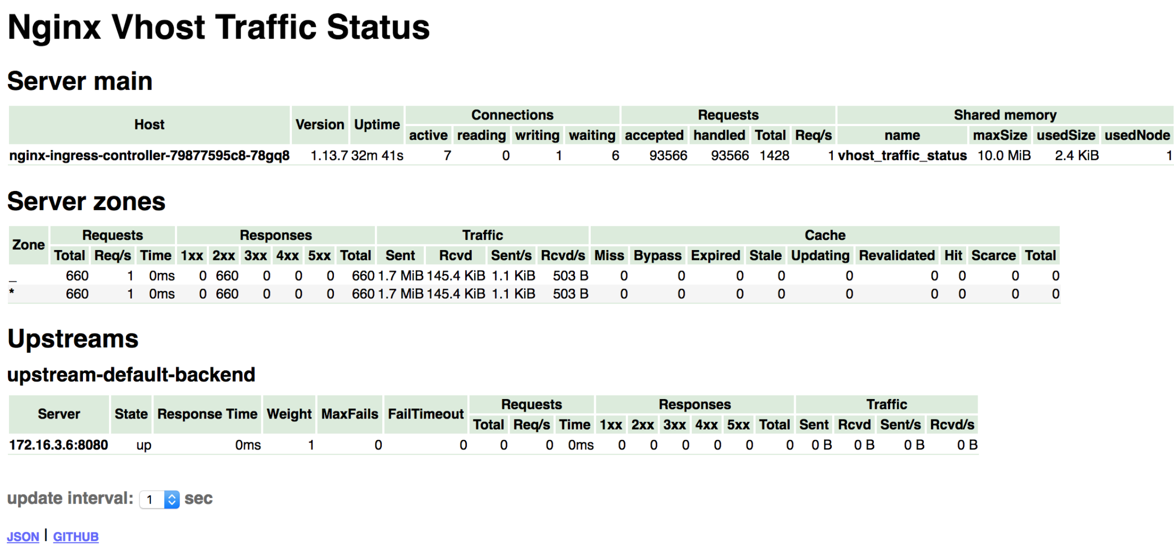
```
root@master # kubectl get pods --selector=app=ingress-nginx -n kube-system
NAME READY STATUS RESTARTS AGE
nginx-ingress-controller-79877595c8-78gq8 1/1 Running 0 1h
root@master # kubectl exec -it nginx-ingress-controller-79877595c8-78gq8 -n kube-system -- cat /etc/nginx/nginx.conf | grep vhost_traffic_status_display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

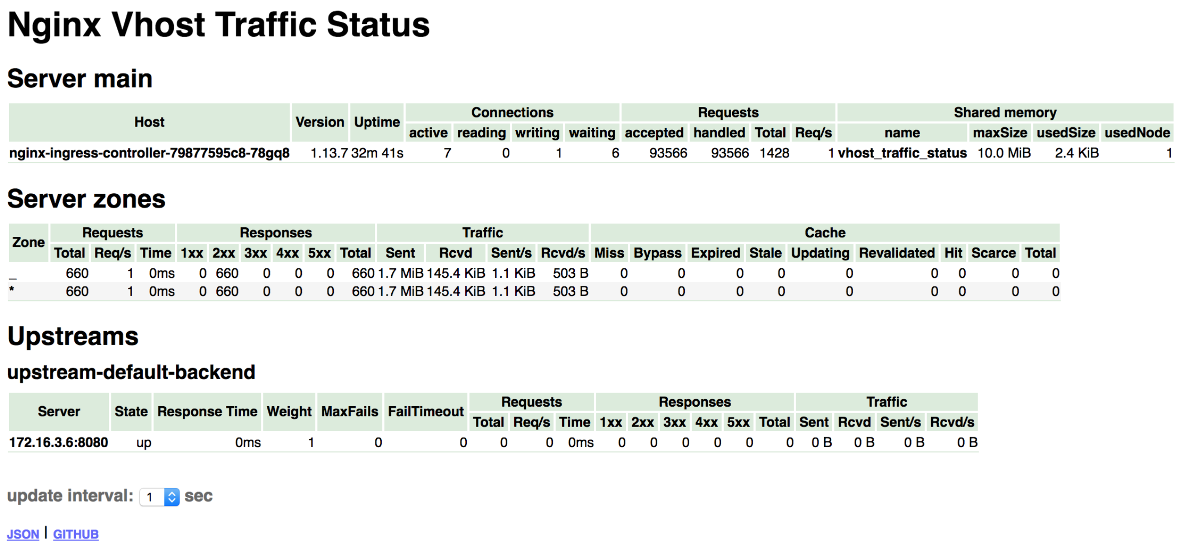
4. Locally access the Ingress Nginx monitoring console.

   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

1.9.5 Ingress configurations

Alibaba Cloud Container Service provides the highly reliable Ingress controller components and integrates with Alibaba Cloud Server Load Balancer to provide the flexible and reliable Ingress service for your Kubernetes clusters.

See the following Ingress orchestration example. You must configure the annotation when creating an Ingress in the Container Service console. Some configurations must create dependencies. For more information, see *Create an Ingress in the Container Service console*, *Support for Ingress*, and *Kubernetes Ingress*. Ingress also supports the configuration of configmap. For more information, see *https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/*.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/service-match: 'new-nginx: header("
foo", /^bar$/)'            #Gray release rule.Header is used in this
example.
    nginx.ingress.kubernetes.io/service-weight: 'new-nginx: 50,old-
nginx: 50'                 #Traffic weight annotation
  creationTimestamp: null
  generation: 1
  name: nginx-ingress
  selfLink: /apis/extensions/v1beta1/namespaces/default/ingresses/
nginx-ingress
spec:
  rules:                                                          ##
Ingress rule
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: new-nginx
```

```
          servicePort: 80
        path: /
      - backend:
          serviceName: old-nginx
          servicePort: 80
        path: /
tls:                                                             ##Enable
TLS to set a secure Ingress.
  - hosts:
    - *.xxxxxx.cn-hangzhou.alicontainer.com
    - foo.bar.com
    secretName: nginx-ingress-secret                              ##Secret
 name
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.

1.9.6 Create an Ingress in the Container Service console

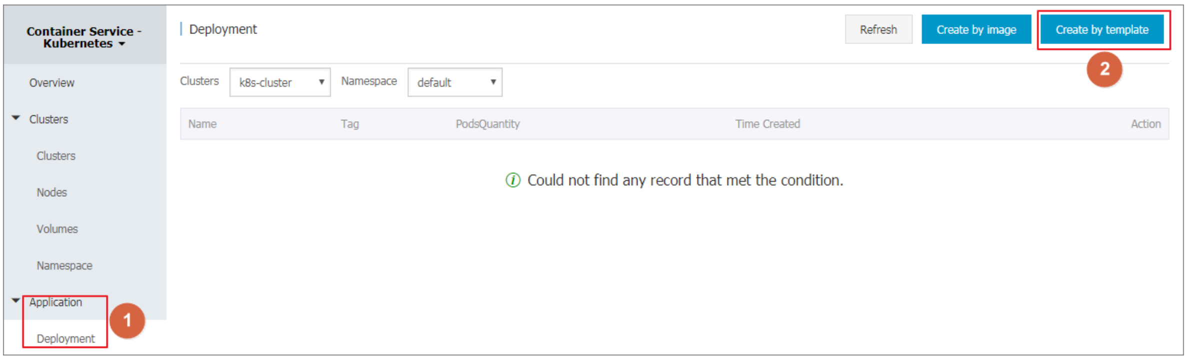
Alibaba Cloud Container Service console integrates with the Ingress service, which allows you to quickly create an Ingress service in the Container Service console to build the flexible and reliable traffic access layer.

Prerequisites

- You have successfully created a Kubernetes cluster and Ingress controller is running normally in the cluster. For how to create a Kubernetes cluster, see *Create a Kubernetes cluster*.
- Log on to the master node by using SSH. For more information, see *Access Kubernetes clusters by using SSH*.

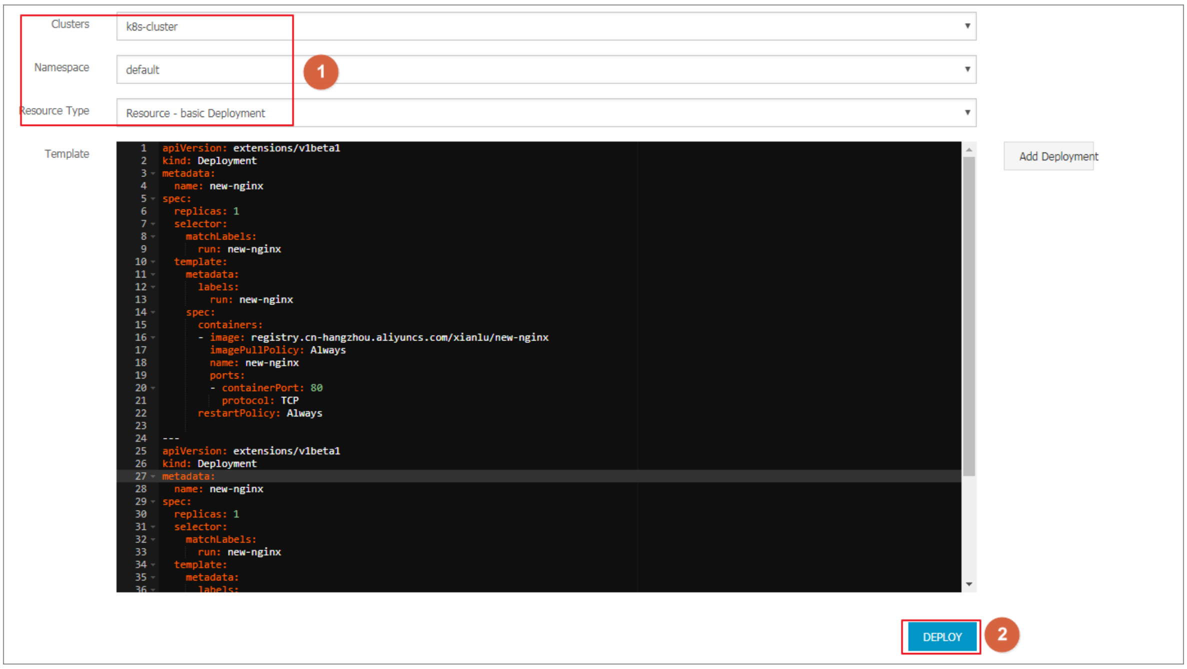
Step 1. Create a deployment and a service

1. Log on to the *Container Service console*.
2. 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.



4. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click DEPLOY.

   In this example, three nginx applications are created. One for the old application (old-nginx), one for the new (new-nginx), and an application for testing the cluster access domain name (domain-nginx).



The orchestration template for old-nginx is as follows:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: old-nginx
spec:
  replicas: 2
  selector:
    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
```

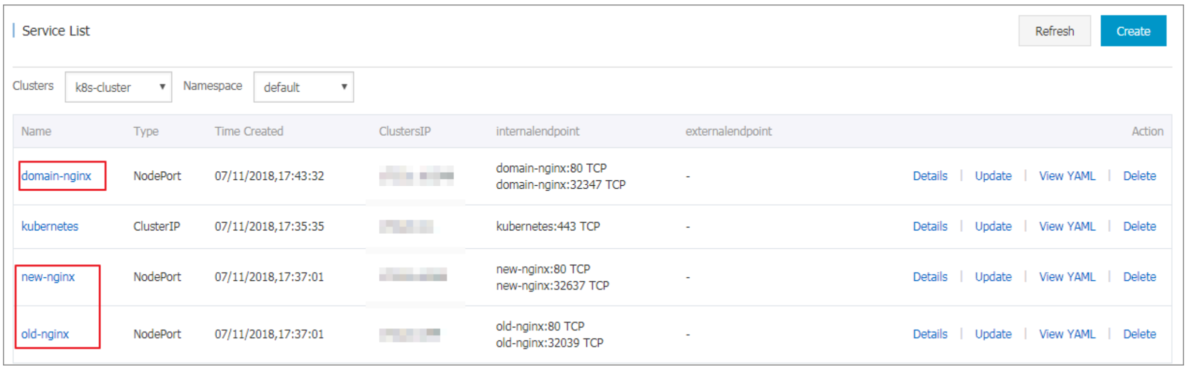
The orchestration template for domain-nginx is as follows:

```
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
```

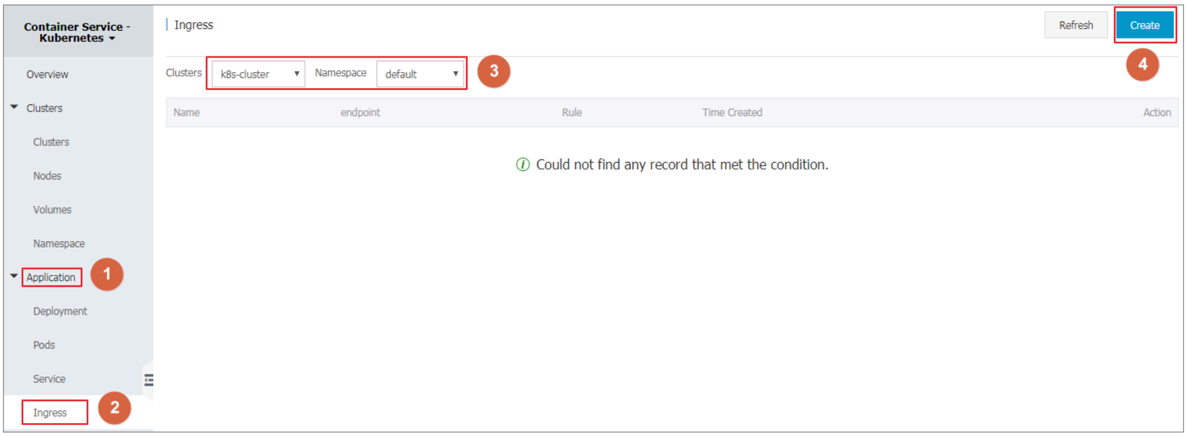
5. Click Application > Service in the left-side navigation pane to enter the Services List page.

   After the service is created, you can see it on the Service List page.

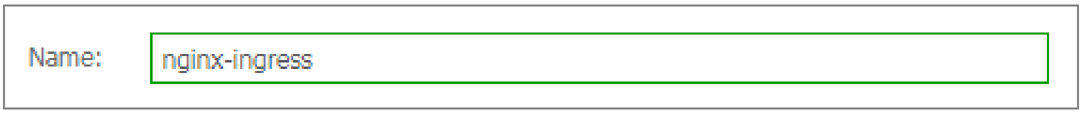


Step 2. Create an Ingress

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.



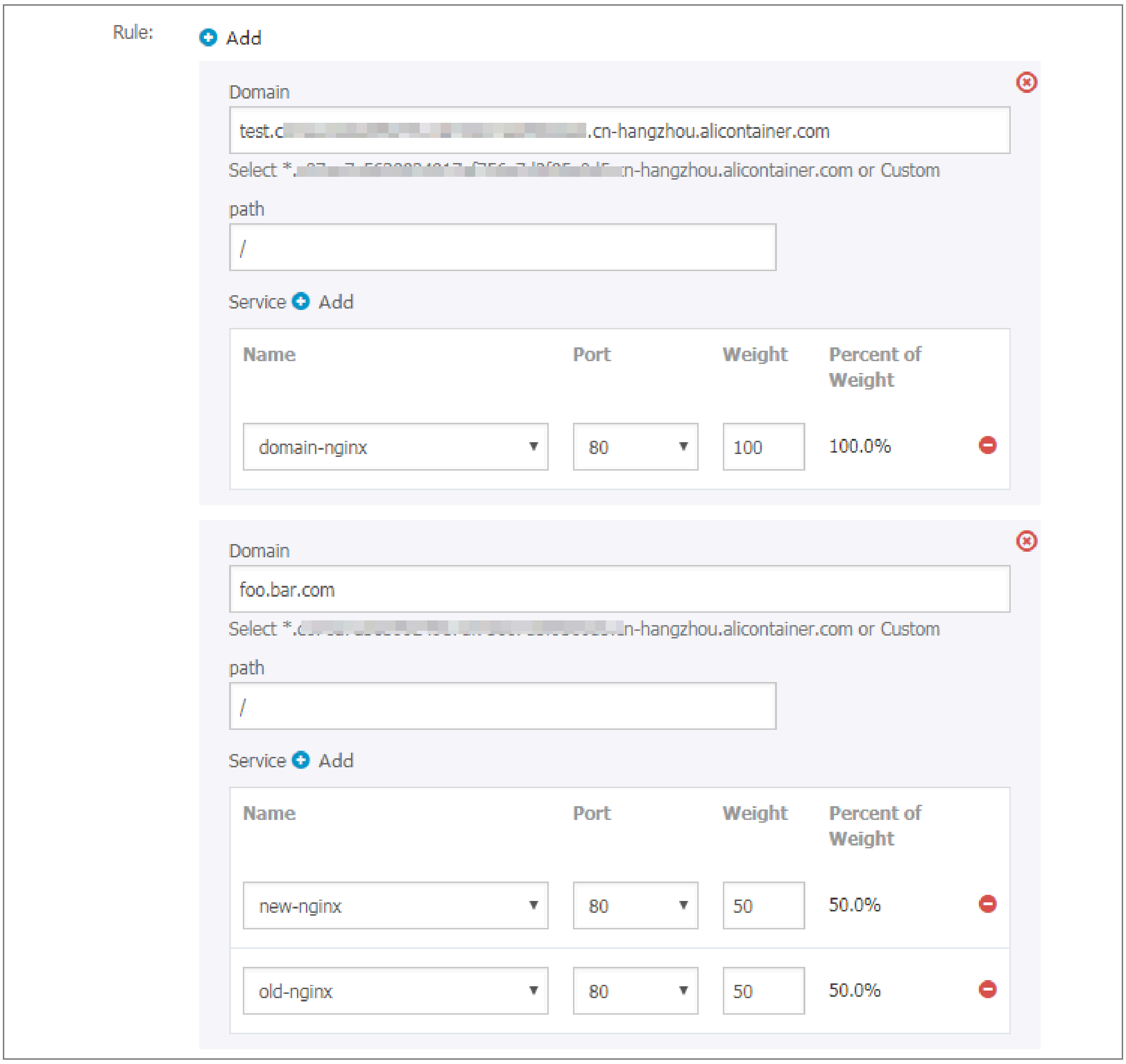
4. In the displayed dialog box, enter the Ingress name. In this example, enter nginx-ingress.



5. Configure the rules.

   The Ingress rules are the rules that authorize the inbound access to the cluster and are generally the HTTP rules. Configure the domain name (virtual hostname), URL path, service name, and port. For more information, see *Ingress configurations*.

   In this example, add a complicated Ingress rule. Configure the default test domain name and virtual hostname of the cluster to display the Ingress service based on the domain names.



- The simple Ingress based on the default domain name, that is, provide the access service externally by using the default domain name of the cluster.
  - Domain: Enter the default domain name of the cluster. In this example, use `test.[cluster-id].[region-id].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.

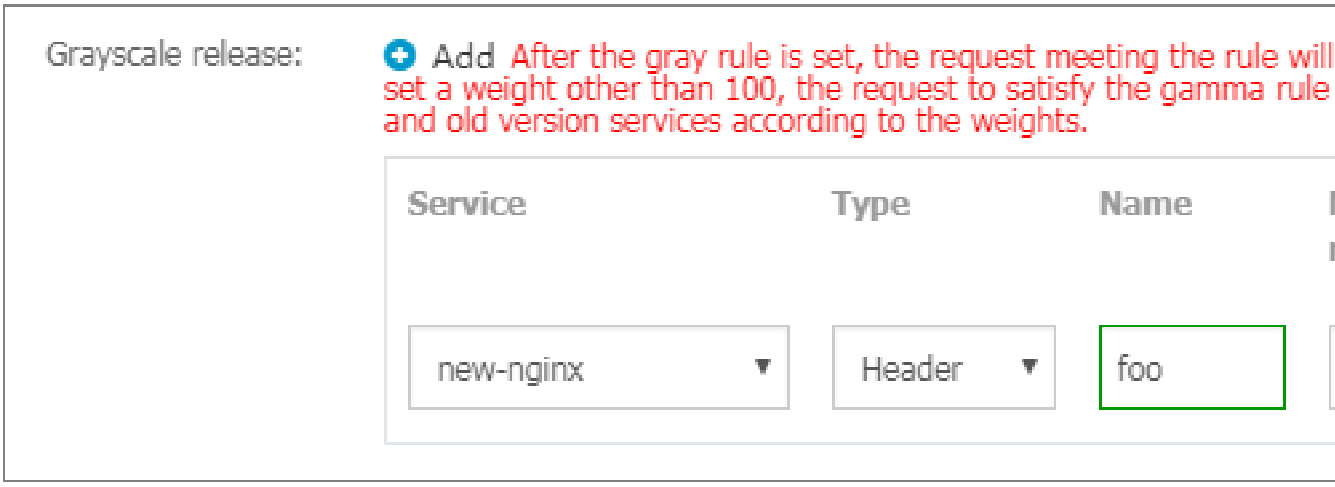
Container Service supports different traffic segmentation methods for grayscale publish and AB test scenarios.

a. Traffic segmentation based on the request header.
b. Traffic segmentation based on cookie.

c. Traffic segmentation based on query (request) parameters.

After the grayscale rule is configured, the request that matches the grayscale publish rule can be routed to the new service version new-nginx. If the service sets a weight rate of less than 100%, requests that match the grayscale publish rule continue to be routed to the corresponding service based on the weight rate.

In this case, set the request header to meet a grayscale publish rule of `foo=^bar$`, only requests with the request header can access the new-nginx service.



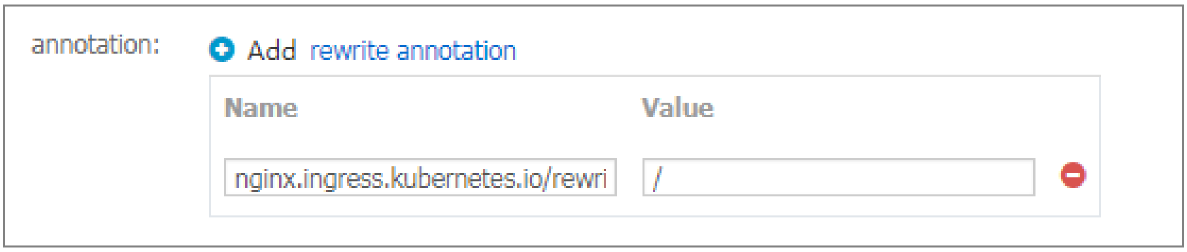
- Service: Routing rule configuration service.
- Type: matching request header, cookie, and query (request) parameters are supported.
- Name and match value: User-defined request field, name and match value are key-value pairs.
- Match rules: Regular and exact matches are supported.

7. Configure the annotations.

Click rewrite annotation, a typical redirection annotation can be added to the route. `nginx.ingress.kubernetes.io/rewrite-target: /` indicates that the `/path` is redirected to the root path `/` that the backend service can recognize.

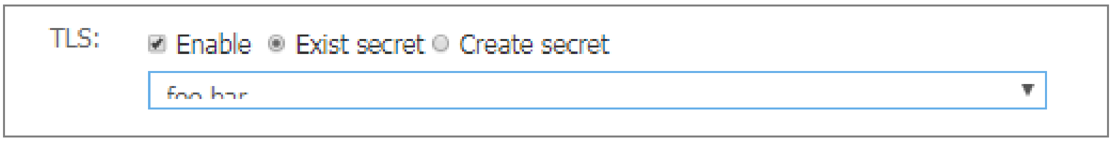
Note:
In this example, the access path is not configured, so no need to configure rewrite annotations. The purpose of the rewrite annotation is to enable Ingress to forward to the backend as the root path, avoiding 404 errors caused by incorrect access path configuration.

You can also click Add to enter the annotation name and value, which is the annotation key-value pair for Ingress. For more information, see *https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/*.



8. Configure TLS. Select Enable and configure the secure Ingress service. For more information, see *Configure a safe routing service*.

   · You can select to use an existing secret.



   a. Log on to the master node and create `tls.key` and `tls.crt`.

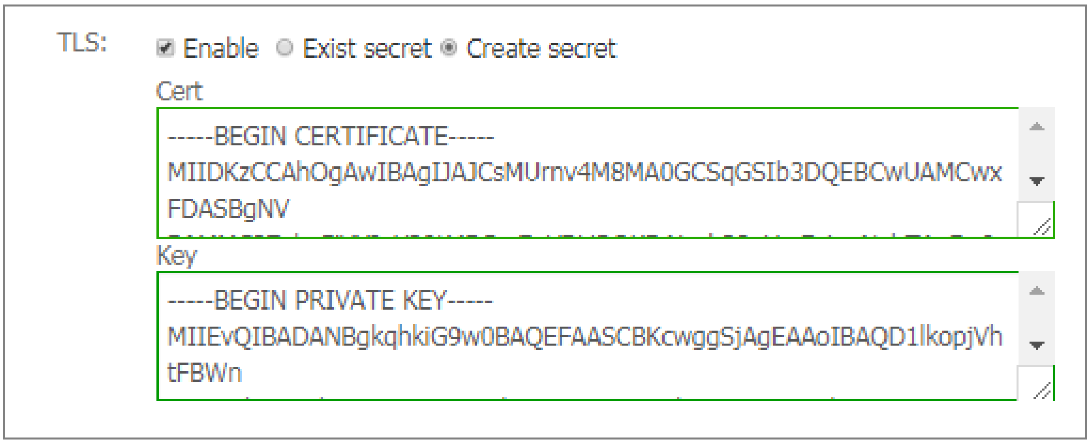
```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls
.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
```

   b. Create a secret.

```
kubectl create secret tls foo.bar --key tls.key --cert tls.crt
```

   c. Run the `kubectl get secret` command to see that secret has been successfully created. You can use the secret that you have created in the Web interface, `foo.bar`.

   · You can create the secret with one click by using the created TLS private key and certificate.



a. Log on to the master node and create `tls.key` and `tls.crt`.

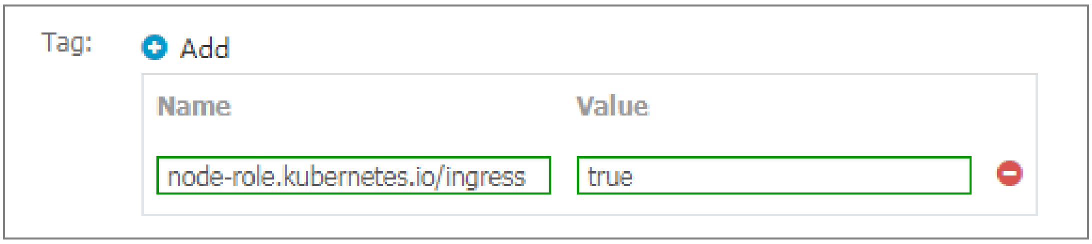
```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls
.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
```

b. Run the `vim tls.key` and `vim tls.crt` to get the generated private key and certificate.

c. Copy the generated certificate and private key to the Cert and Key fields.

9. Adding the tags.

   Add the corresponding tags for Ingress to indicate the characteristics of the Ingress.



10. Click Create.

   The Ingress nginx-ingress is displayed on the Ingress page.

11. Click on the access domain name `test.[cluster-id].[region-id].alicontainer.com` in the route, and `foo.bar.com` to access the welcome page of nginx.

   Click on the route address pointing the new-nginx service and find the page that points the old-nginx application.

Note:

Access the route address in the browser. By default, the request header does not have the `foo=^bar$`, so the traffic is directed to the old-nginx application.

12. Log on to the master node by using SSH. Run the following command to simulate the access result with a specific request header.

```
curl -H "Host: foo.bar.com" http://47.107.20.35
old
 curl -H "Host: foo.bar.com" http://47.107.20.35
old
 curl -H "Host: foo.bar.com" http://47.107.20.35 # Similar to
browser access requests
old
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.20.35 #
Simulate an access request with a unique header, returning results
based on routing weight
new
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.20.35
old
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.20.35
old
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.20.35
new
```

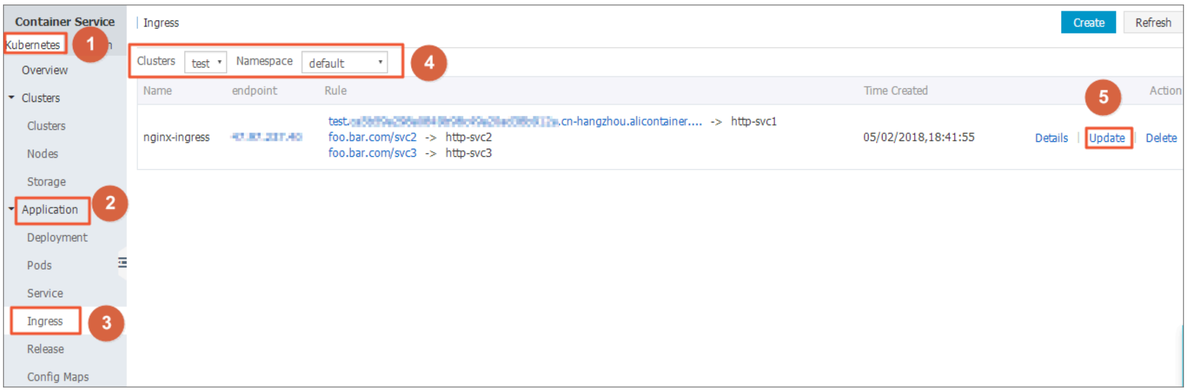
1.9.7 Update an Ingress

Prerequisites

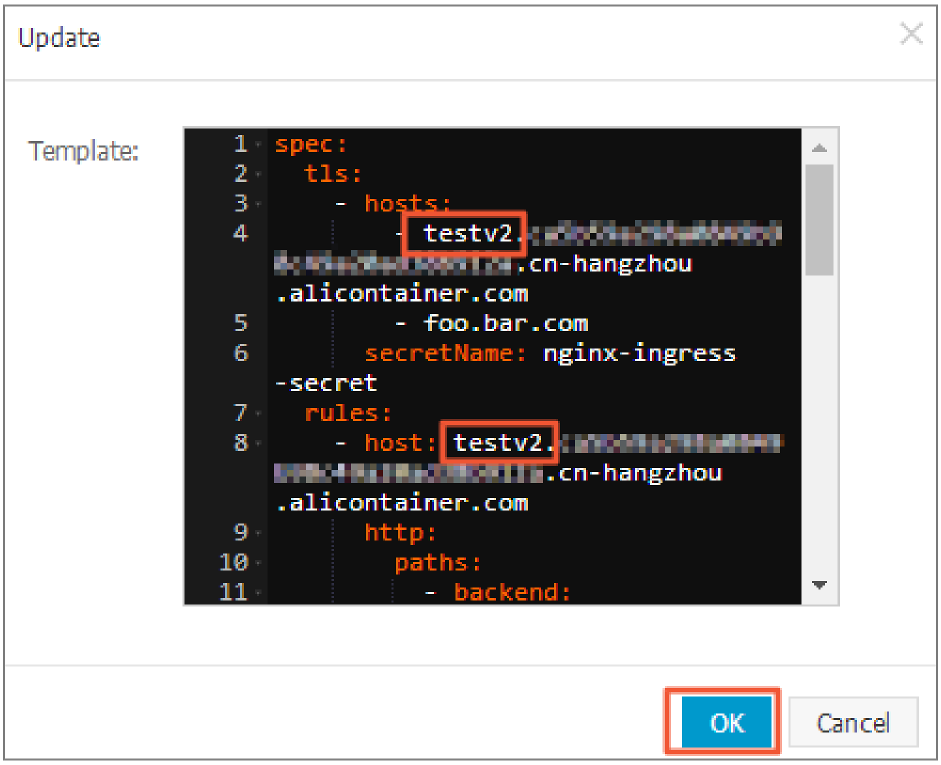
- You have successfully created a Kubernetes cluster and Ingress controller is running normally in the cluster. For how to create a Kubernetes cluster, see *Create a Kubernetes cluster*.
- You have successfully created an Ingress. For more information, see *Create an Ingress in the Container Service console*.

Procedure

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

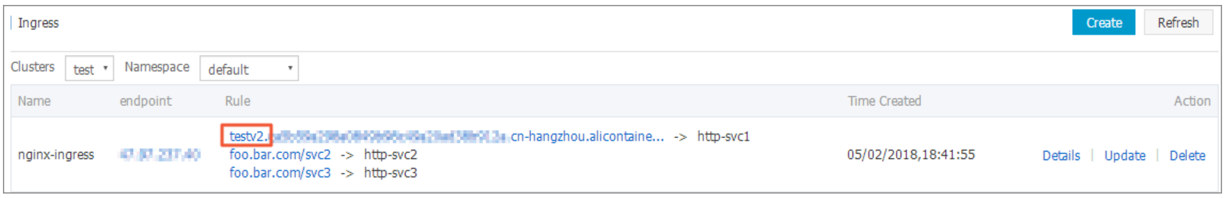


4. Update the Ingress parameters in the displayed dialog box and then click OK. change `test.[cluster-id].[region-id].alicontainer.com` to `testv2.[cluster-id].[region-id].alicontainer.com`。



What's next

On the Ingress page, you can see a rule of this Ingress is changed.



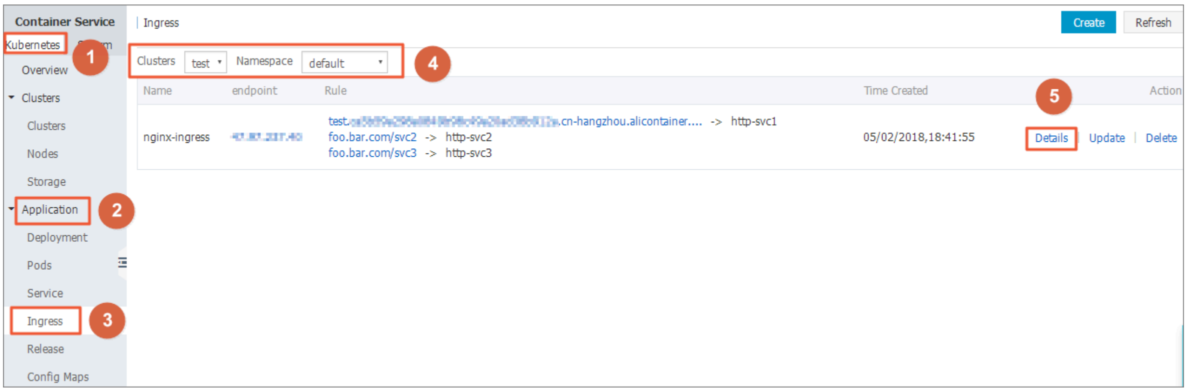
1.9.8 View Ingress details

Prerequisites

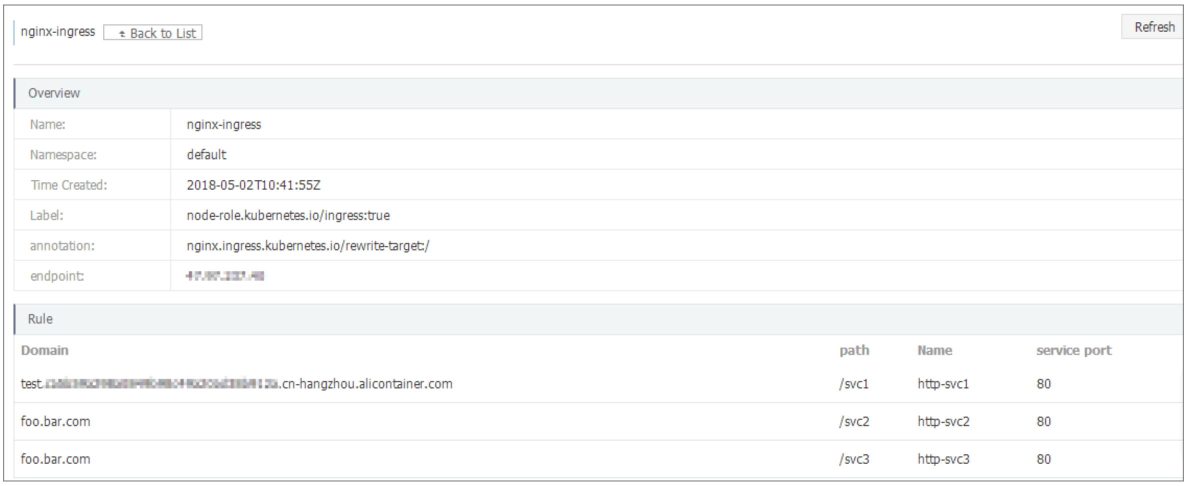
- You have successfully created a Kubernetes cluster and Ingress controller is running normally in the cluster. For how to create a Kubernetes cluster, see *Create a Kubernetes cluster*.
- You have successfully created an Ingress. For more information, see *Create an Ingress in the Container Service console*.

Procedure

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



On the details page, you can view the overview and rules of the Ingress.



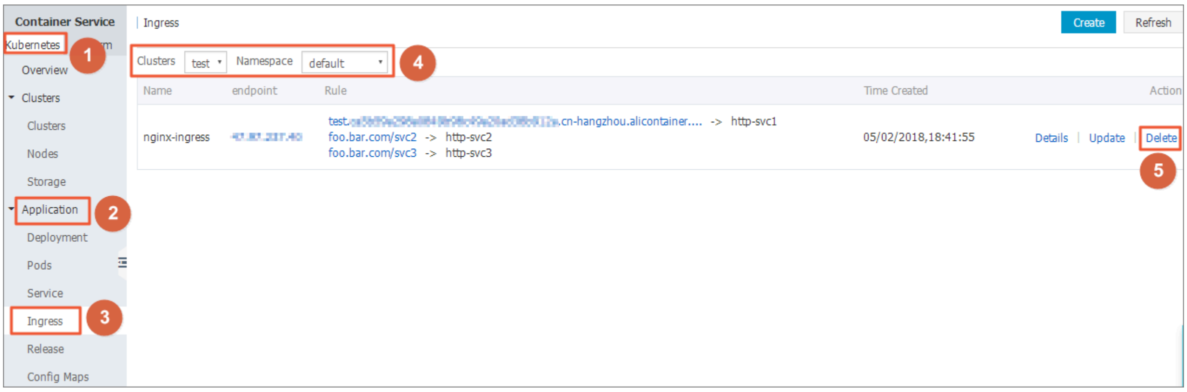
1.9.9 Deleting a route

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 the Container Service console*.

Procedure

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



4. Click Confirm in the displayed dialog box.

1.10 Config map and Secret management

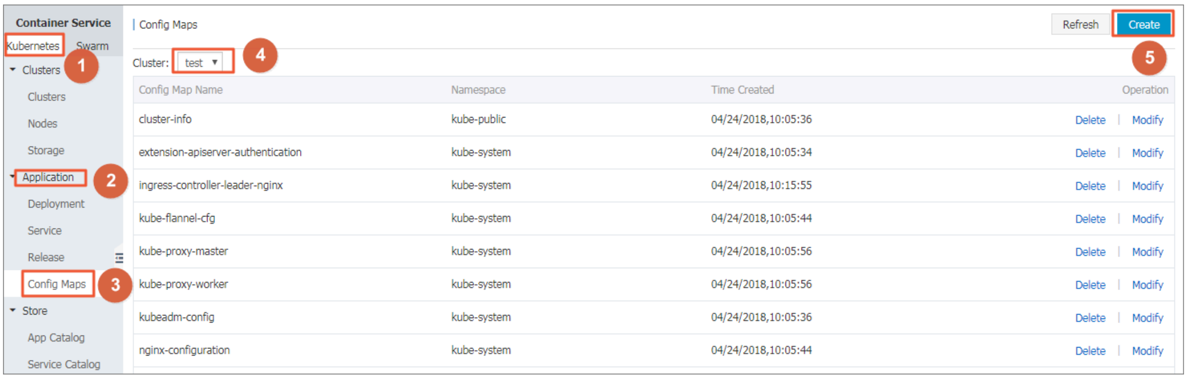
1.10.1 Create a config map

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

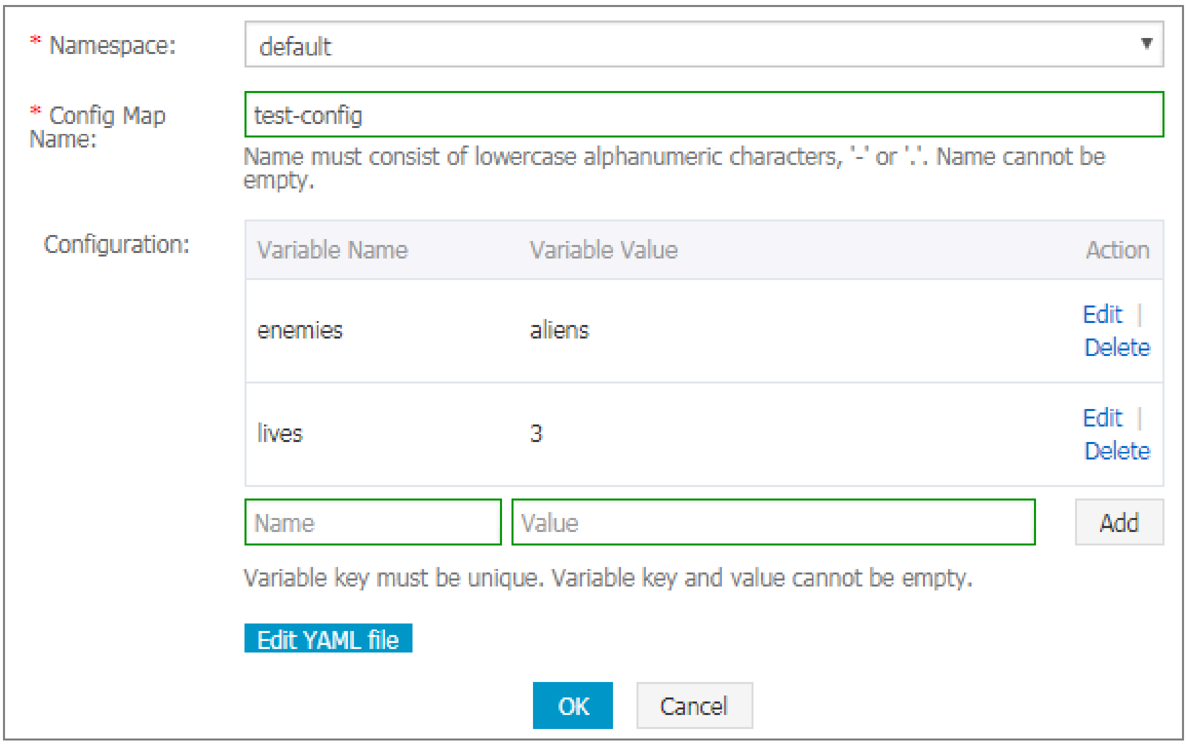
1. Log on to the *Container Service console*.

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.



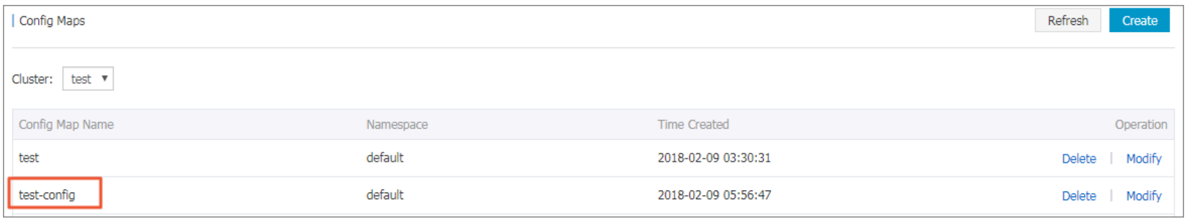
4. Complete the settings and then click OK.

   - Namespace: Select the namespace to which the config map belongs. Config map is a Kubernetes resource object that must be applied to the namespace.
   - Config Map Name: Enter the config map name, which can contain lowercase letters, numbers, hyphens (-), and periods (.). The name cannot be empty. Other resource objects must reference the config map name to obtain the configuration information.
   - Configuration: Enter the Variable Name and the Variable Value. Then, click Add on the right. You can also click Edit, complete the configuration in the displayed dialog box, and click OK.



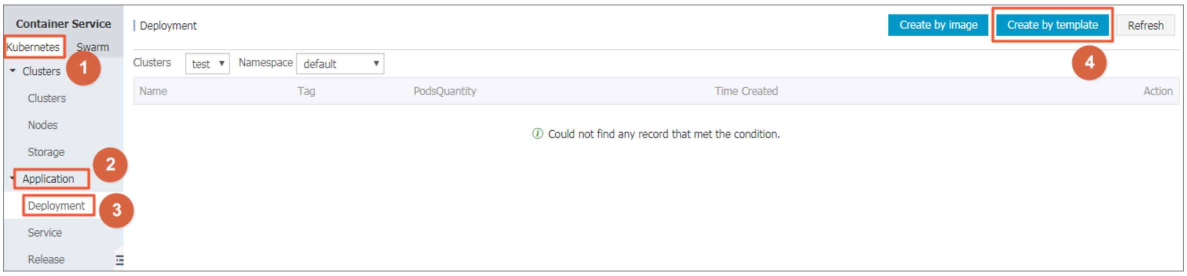
In this example, configure the variables enemies and lives to pass the parameters aliens and 3 respectively.

5. You can view the config map test-config on the Config Maps page after clicking OK.

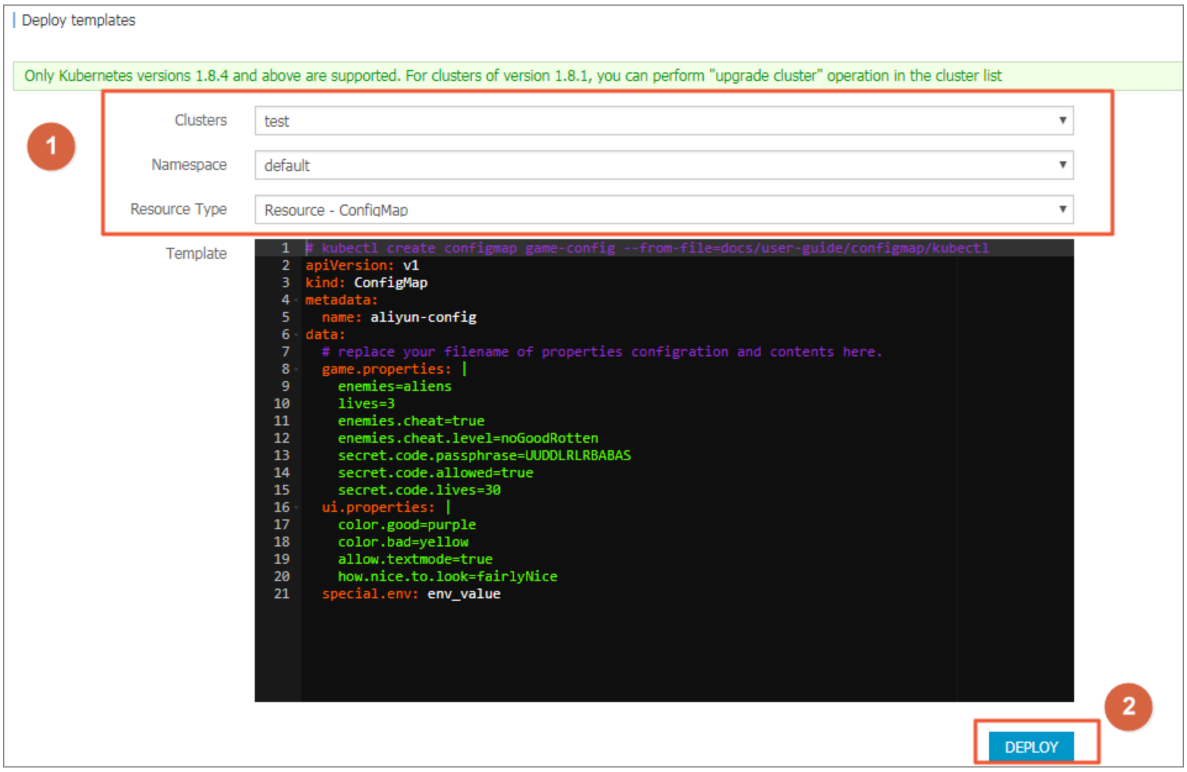


Create a config map by using a template

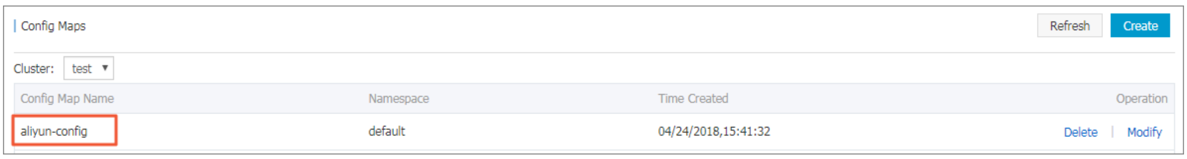
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Application > Deployment in the left-side navigation pane.
3. Click Create by template in the upper-right corner.



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.



5. After the successful deployment, you can view the config map aliyun-config on the Config Maps page.



1.10.2 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 > DeploymentClick 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.

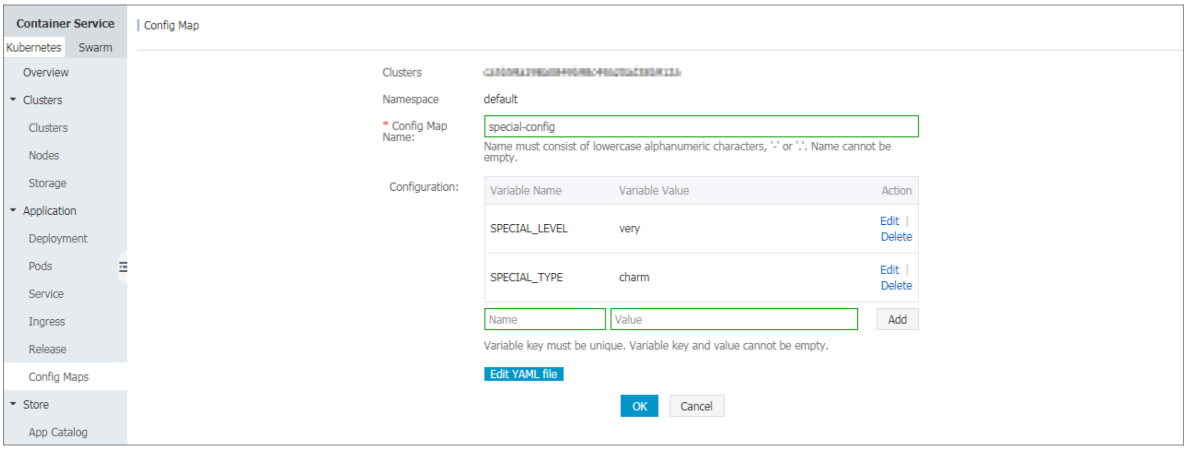
   You can use the following YAML sample template to create a config map.

```
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, clickApplication > 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.



Use a config map to define pod environment variables

Use config map data to define pod environment variables

1. Log on to the *Container Service console*.
2. Under Kubernetes, clickApplication > Deployment Click Create by template in the upper-right corner.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click DEPLOY.

   You can define the environment variables in a pod. Use `valueFrom` to reference the value of SPECIAL_LEVEL to define the pod environment variables.

   See the following orchestration example:

```
apiVersion: v1
kind: Pod
metadata:
   name: config-pod-1
spec:
   containers:
     - name: test-container
       image: busybox
       command: [ "/bin/sh", "-c", "env" ]
       env:
         - name: SPECIAL_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 > DeploymentClick 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
```

1.10.3 Update a config map

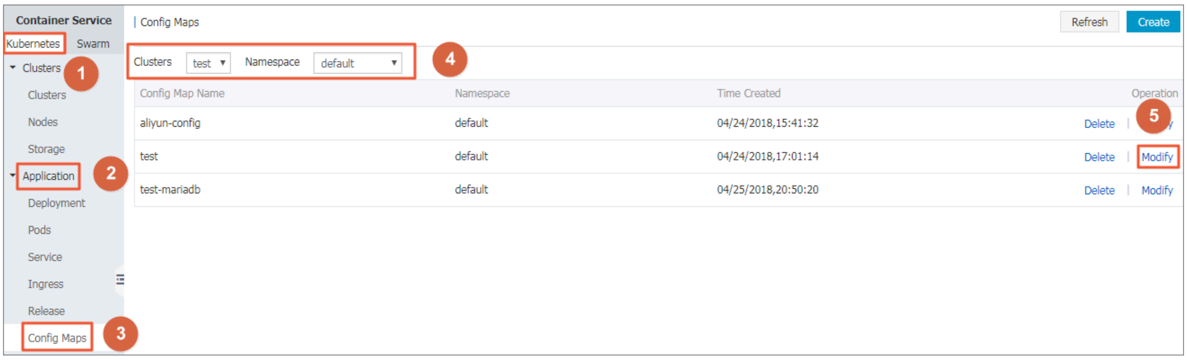
You can modify the configurations of a config map.

Note:

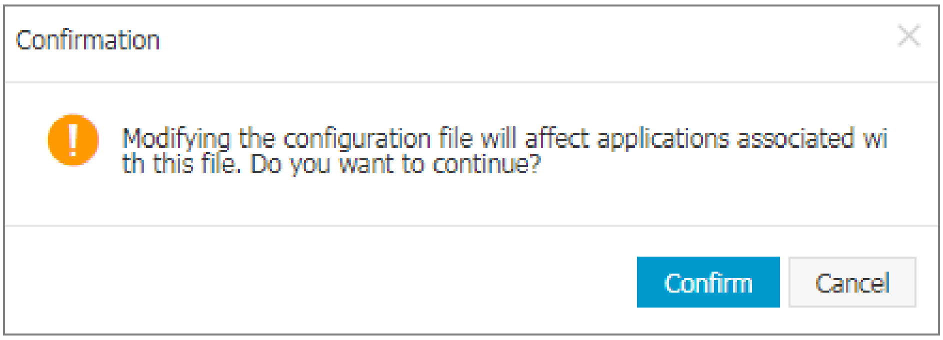
Updating a config map affects applications that use this config map.

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.

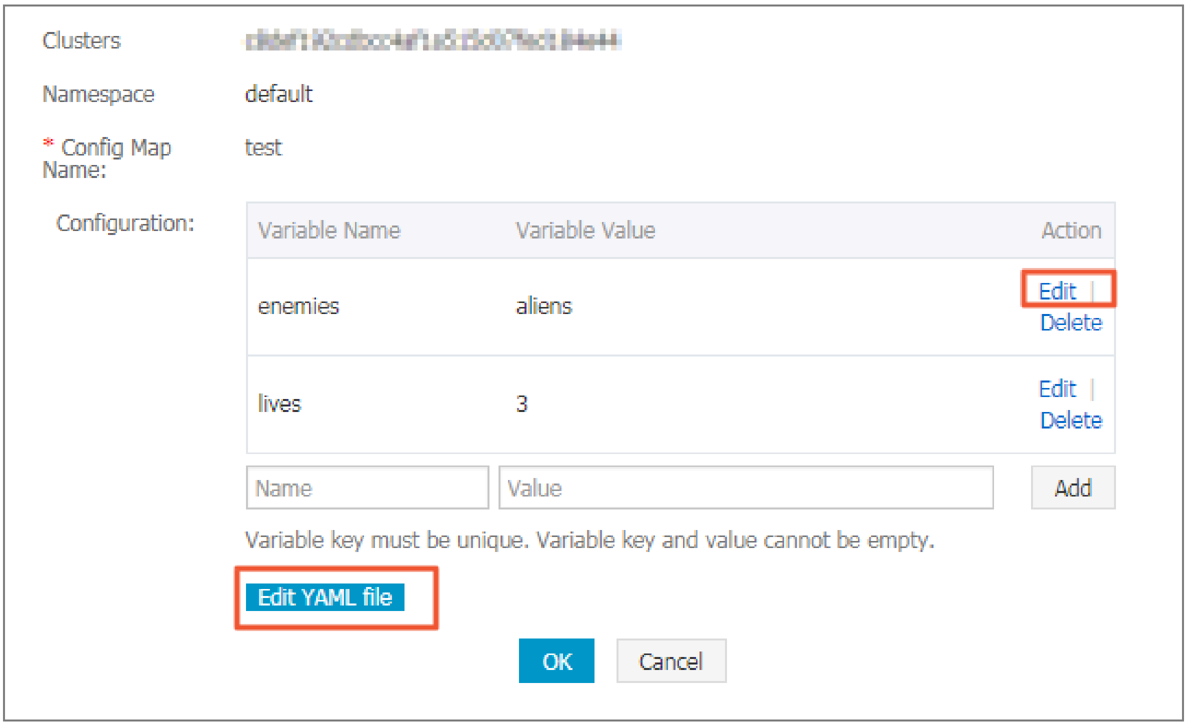


4. Click Confirm in the displayed dialog box.



5. Modify the configurations.

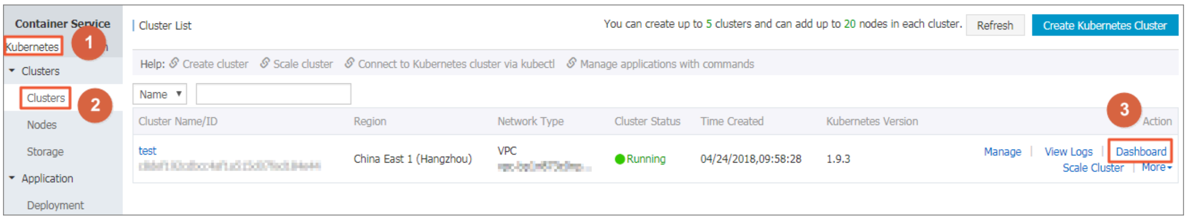
   - Click Edit on the right of the configuration you want to modify. Update the configuration and then click Save.
   - You can also click Edit YAML file. Click OK after making the modifications.



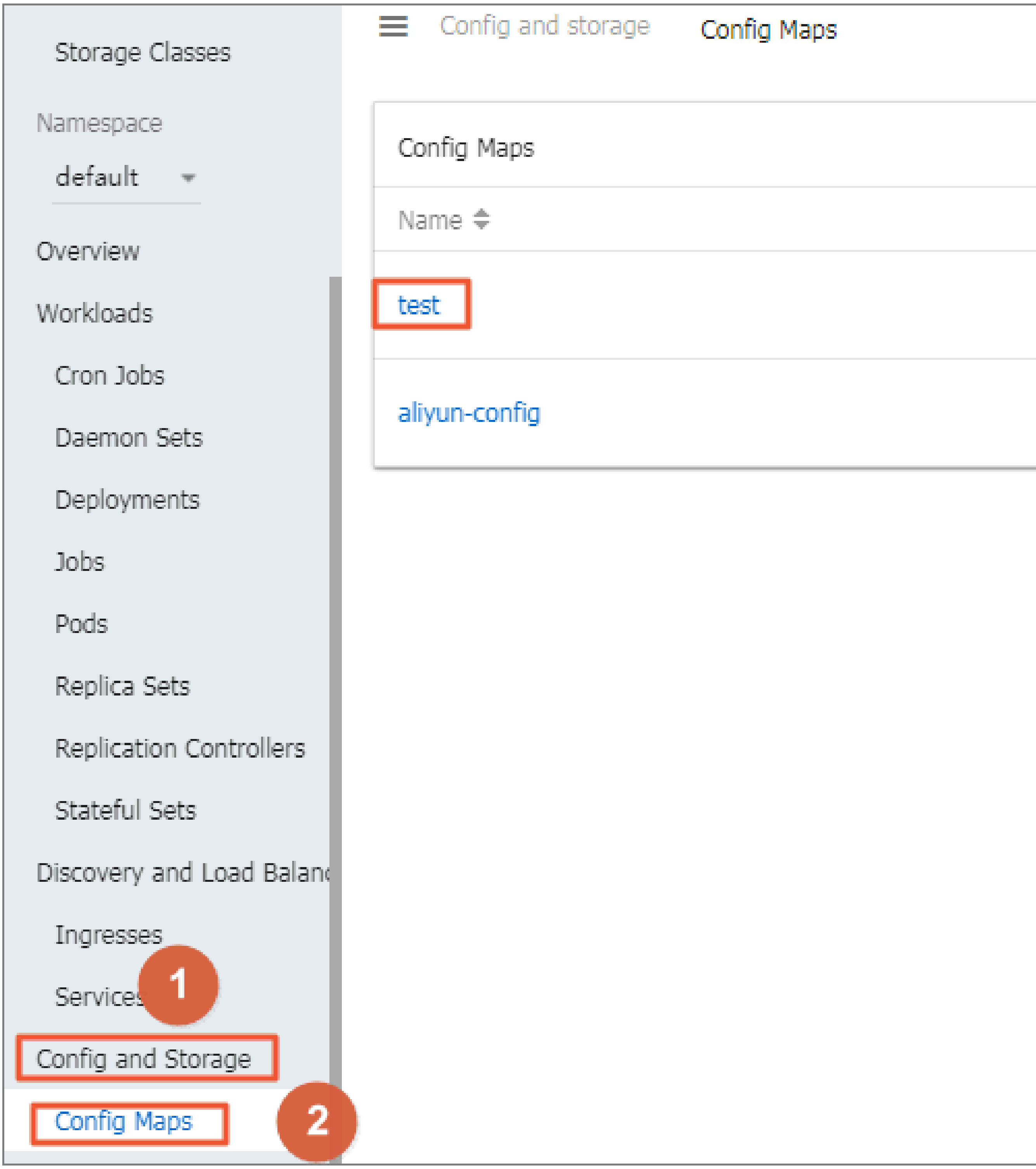
6. After modifying the configurations, click OK.

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 Dashboardat the right of the cluster.



4. Under Kubernetes, select a namespace, click Config and Storage > Secrets in the left-side navigation pane. Select the target secret and clickActions > View/edit YAML.



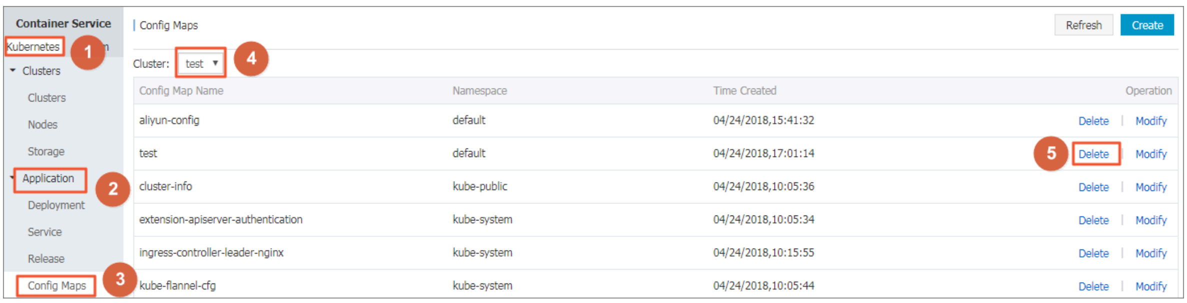
5. The Edit a Secret dialog box appears. Modify the configurations and then click UPDATE.

1.10.4 Delete a config map

You can delete a config map that is no longer in use.

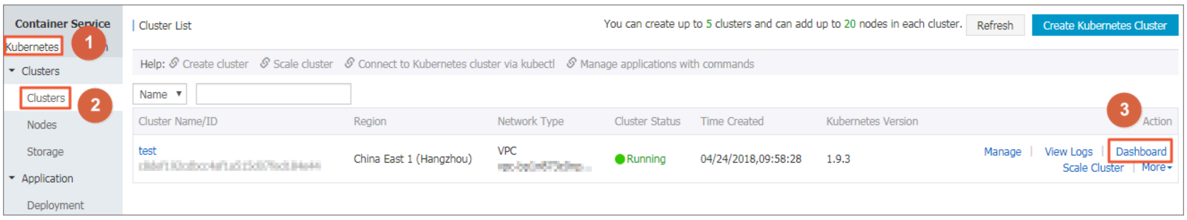
Delete a config map on Config Maps page

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.

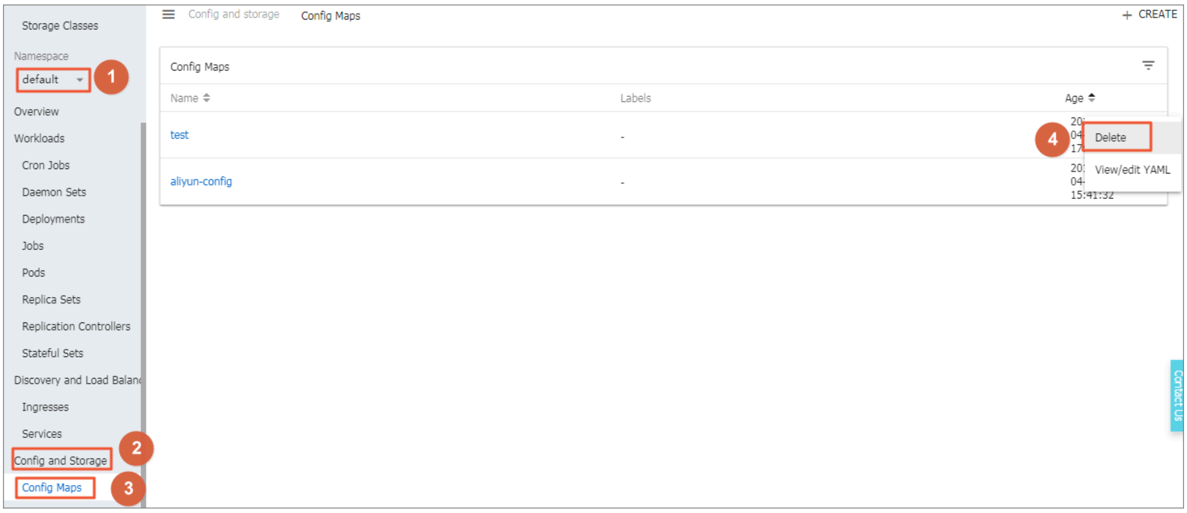


Delete a config map in Kubernetes dashboard

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.



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.



5. Click Delete in the displayed dialog box.

1.10.5 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:

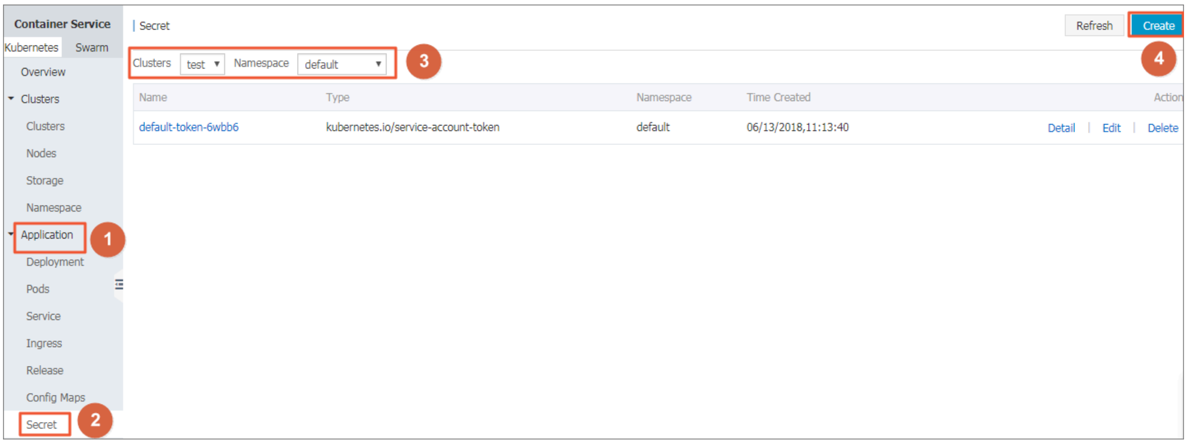
- Service Account: Automatically created by Kubernetes, which is used to access Kubernetes APIs and is automatically mounted to the pod directory `/run/secrets/kubernetes.io/serviceaccount`.
- Opaque: Secret in the base64 encoding format, which is used to store sensitive information such as passwords 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* .

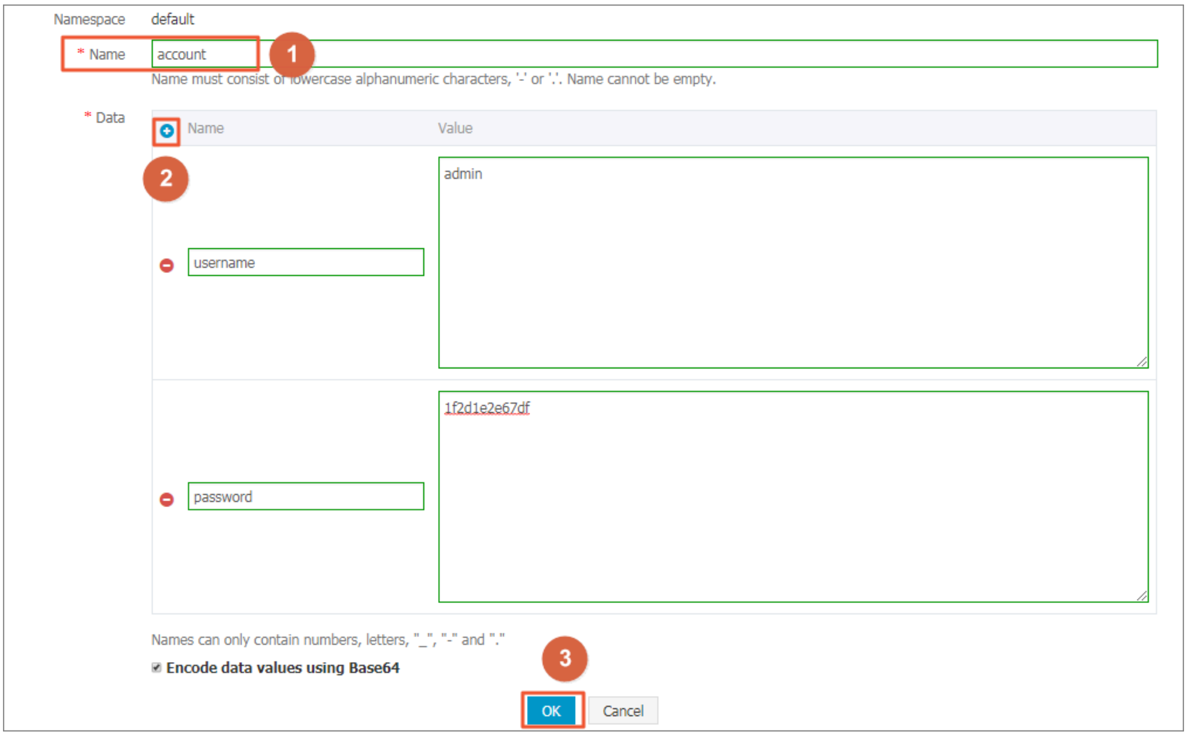
Procedure

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.



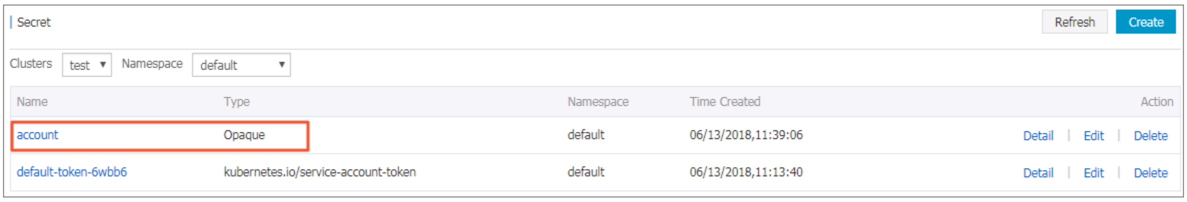
4. Complete the configurations to create a secret.

> Note:
> To enter the clear data of the secret, select the Encode data values using Base64 check box.



a. Name: Enter the secret name, which must be 1–253 characters long, and can only contain lowercase letters, numbers, hyphens (-), and dots (.).
b. Configure the secret data. Click the add icon next to Name and enter the name and value of the secret, namely, the key-value pair. In this example, the secret contains two values: `username:admin` and`passwrod: 1f2d1e2e67df`.
c. Click OK.

5. The Secret page appears. You can view the created secret in the secret list.



1.10.6 View secret details

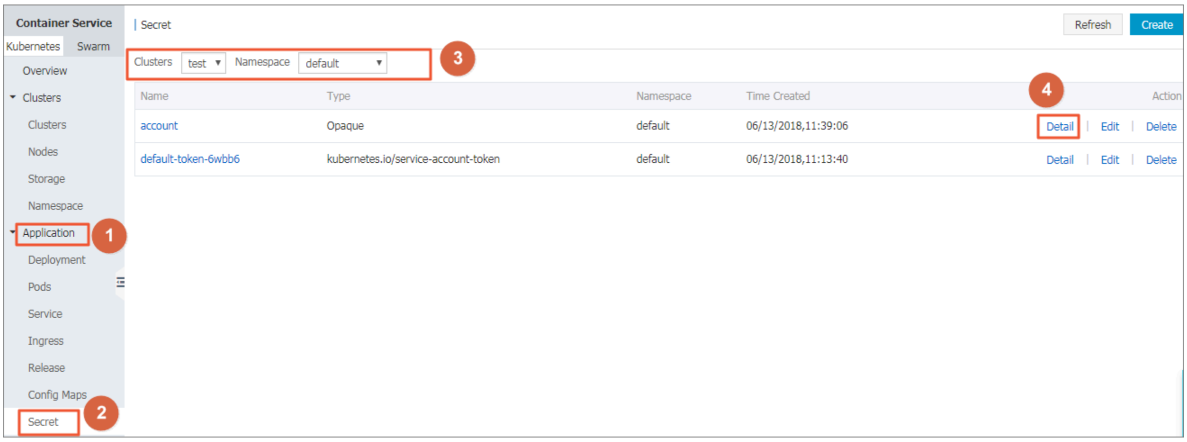
You can view the details of a created secret in the Container Service console.

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*.

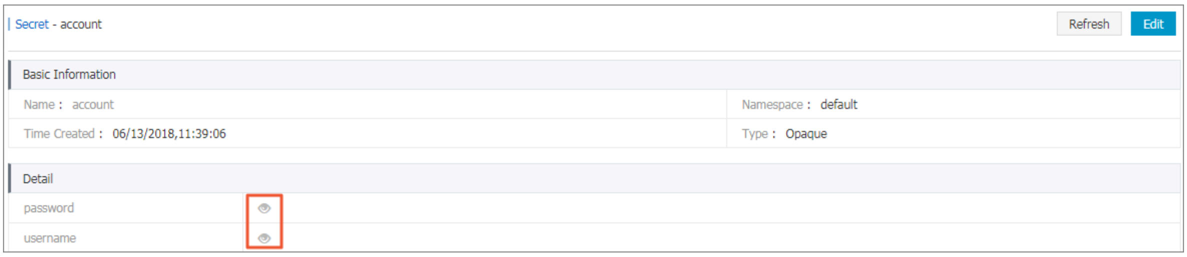
Procedure

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.



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.



1.10.7 Update a secret

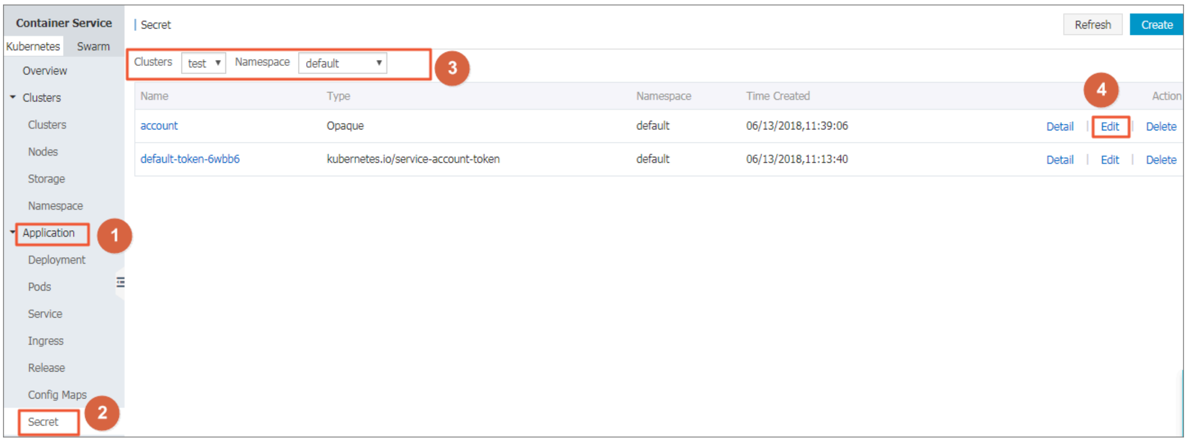
You can update an existing secret directly in the Container Service console.

Prerequisites

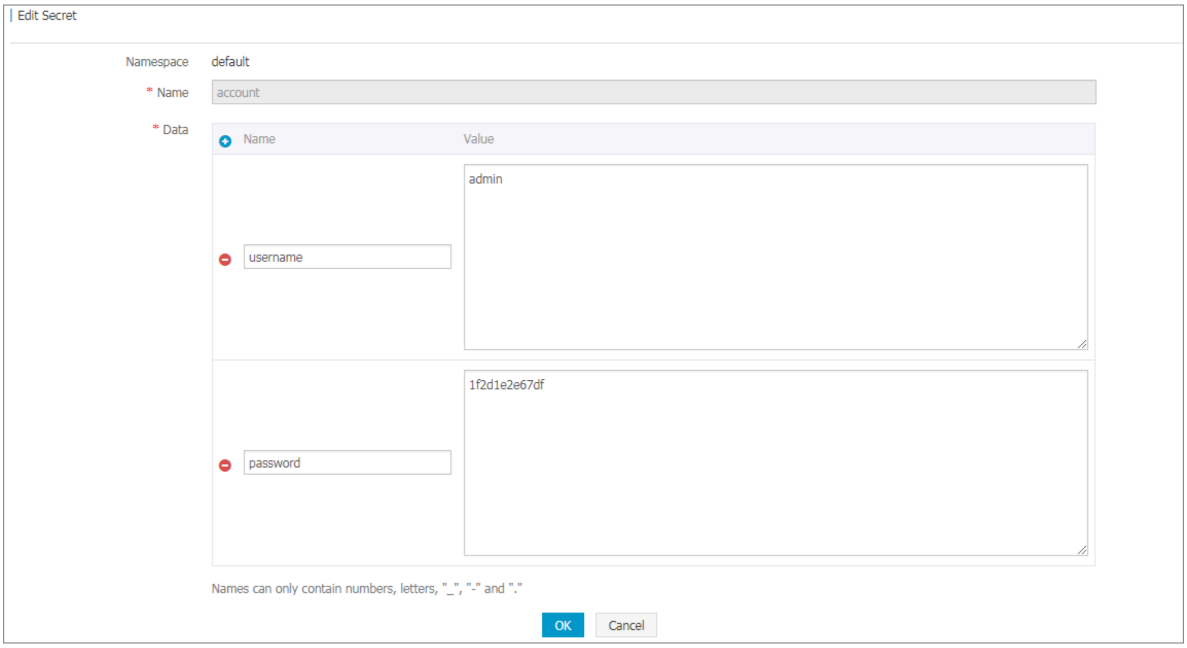
- You have created an Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a secret. For more information, see *Create a secret*.

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.



4. Update the secret data on the Edit Secret page.



5. Click OK.

1.10.8 Delete a secret

You can delete an existing secret directly in the Container Service console.

Prerequisites

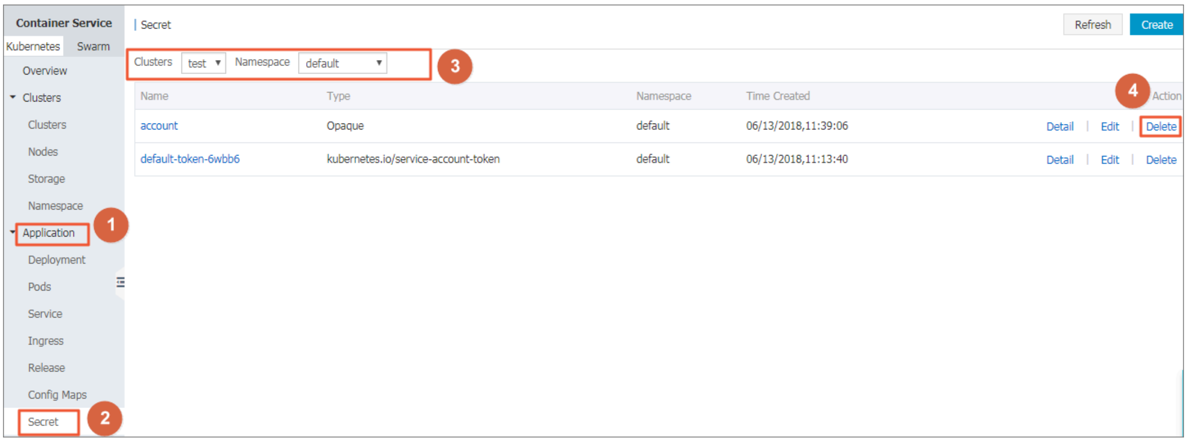
- You have created an Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a secret. For more information, see *Create a secret*.

Context

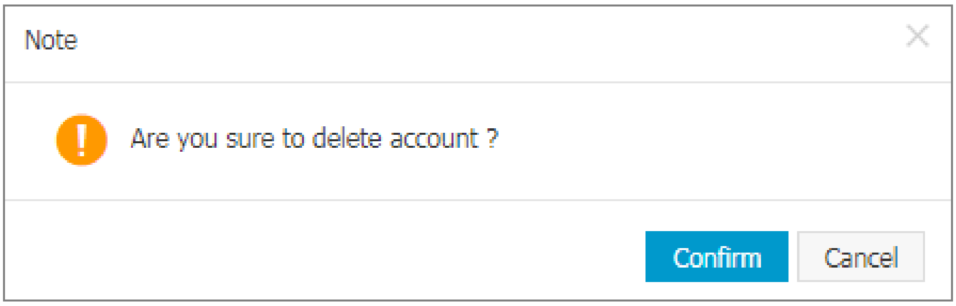
> **Note:**
> Do not delete the secret generated when the cluster is created.

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.



4. Click Confirm in the displayed dialog box.



1.11 Storage management

1.11.1 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.

1.11.2 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/
```

1.11.3 Use Alibaba Cloud cloud disk volumes

You can use Alibaba Cloud cloud disk volumes in a Kubernetes cluster of Alibaba Cloud Container Service.

You can mount an Alibaba Cloud cloud disk 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 cloud disk through a volume.*
  - *Use a 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 disk is 20 GiB.

Static volumes

You can use an Alibaba Cloud cloud disk through a volume or through a PV and PVC.

Prerequisites

You 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 parameter indicates the ID of a mounted cloud disk. The volume name and PV name must be the same as the value of the volumeId parameter.
- In a Kubernetes cluster, a cloud disk can be mounted only to a node that resides in the same zone as the cloud disk.
- Only Pay-As-You-Go cloud disks can be mounted. In a Kubernetes cluster, the ECS instance billing method can be changed to Subscription, but the cloud disk billing method cannot be changed to Subscription. Otherwise, the cloud disks will fail to be mounted.

Use a cloud disk through a volume

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 cloud disk 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

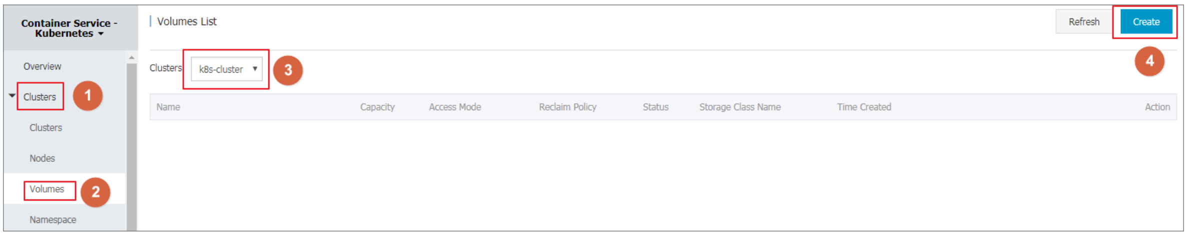
Use the following `disk-pv.yaml` file to create a PV:

Note:
The PV name must be the same as the cloud disk ID.

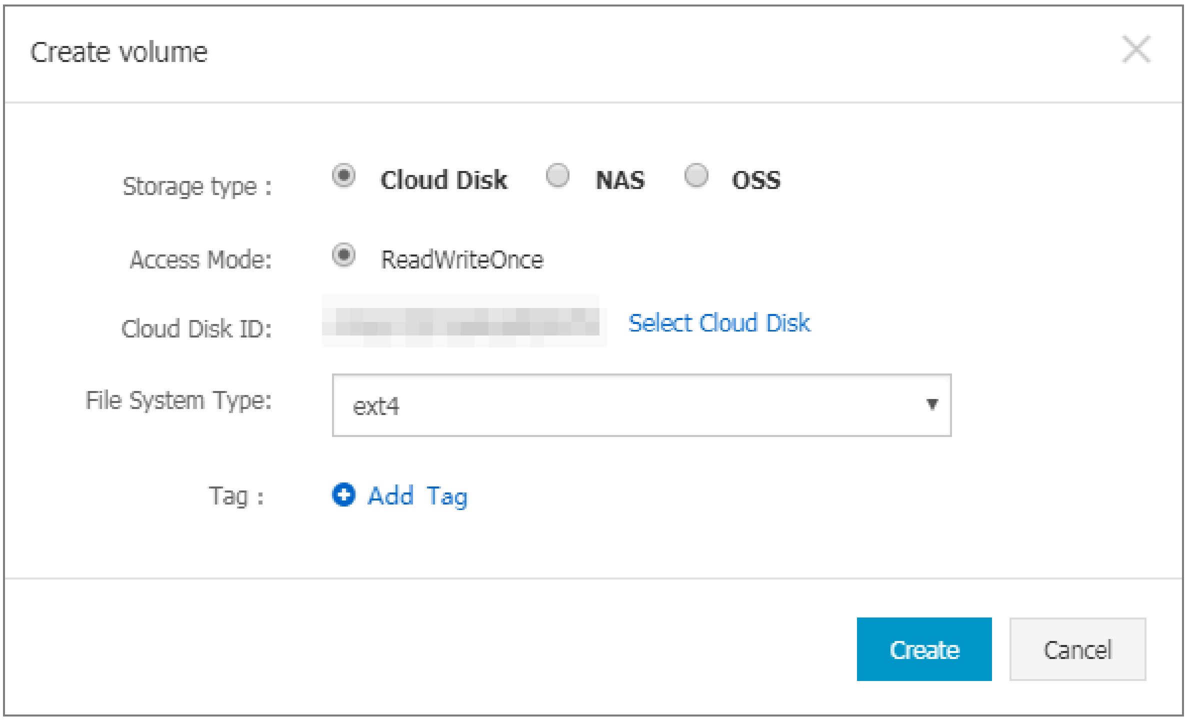
```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: d-bp1j17ifxfasvts3tf40
  labels:
    failure-domain.beta.kubernetes.io/zone: cn-hangzhou-b
    failure-domain.beta.kubernetes.io/region: cn-hangzhou
spec:
  capacity:
    storage: 20Gi
  storageClassName: disk
  accessModes:
    - ReadWriteOnce
  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.



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
reclaimPolicy: Retain
```

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 disk.
- alicloud-disk-ssd, namely, an SSD disk.
- alicloud-disk-available: This StorageClass provides a systematic method of disk selection. Specifically, the system first attempts to create an Ultra disk. If the Ultra disks in the specified zone are sold out, the system tries to create an SSD disk. If the SSD 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 volumeClai mTemplates 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
```

1.11.4 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 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 specify a NAS sub-directory and mount it to your NAS volume. If the NAS sub-directory specified by you does not exist, the system automatically creates the NAS sub-directory and mounts it to your NAS volume.
- 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.

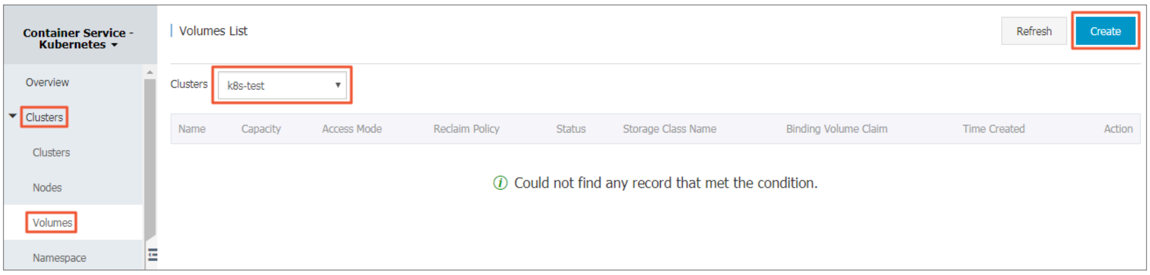
- Create a PV by using a YAML file.

  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"
```

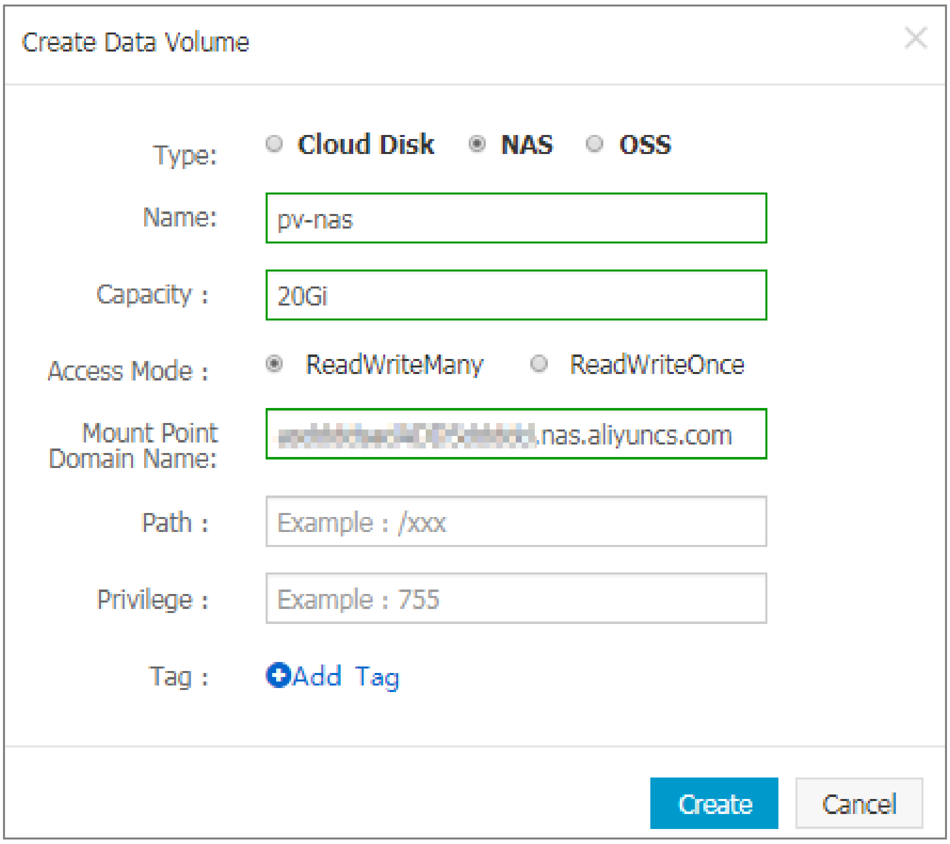
- Create a NAS volume in the Container Service console.

  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.



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.



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
```

1.11.5 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 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

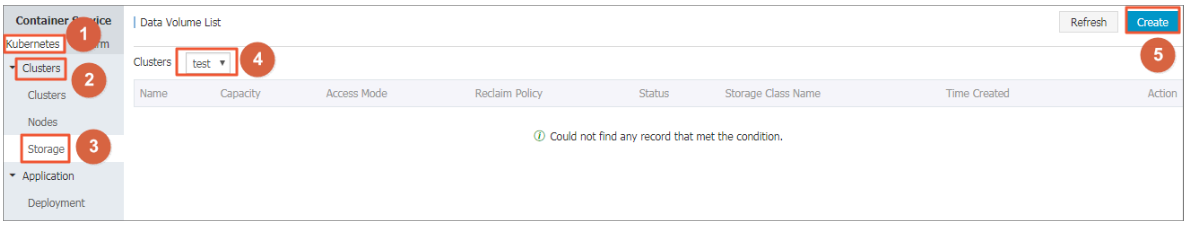
Use a *oss-pv.yaml* file to create a PV as follows:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-oss
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  storageClassName: oss
  flexVolume:
    driver: "alicloud/oss"
```

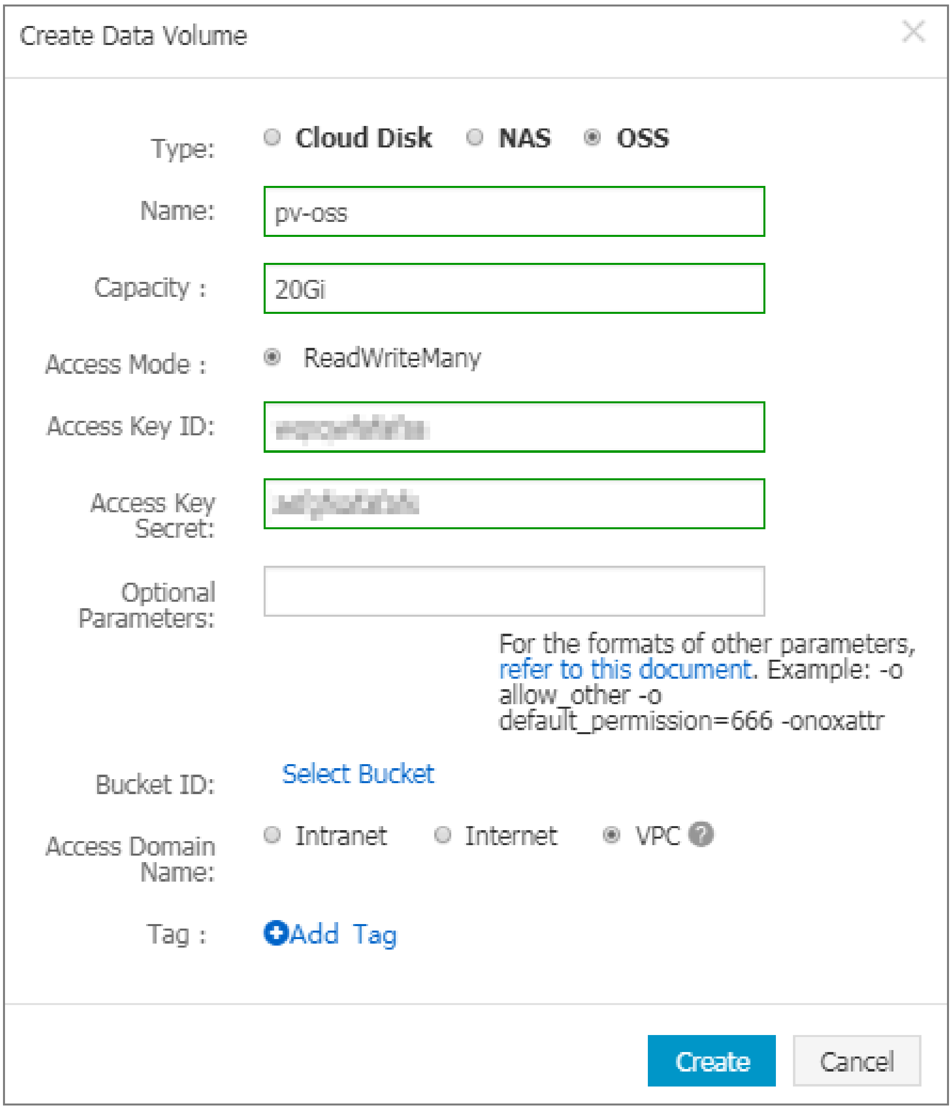
```
    options:
      bucket: "docker"
      url: "oss-cn-hangzhou.aliyuncs.com"
      akId: ***
      akSecret: ***
      otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

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.



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 clickSelect.
   - Access Domain Name. If the selected bucket and the cluster ECS instances are in different regions, you need to selectInternet. 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.



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.

1.11.6 Create a persistent storage volume claim

You can create a persistent storage volume claim (PVC) by using the Container Service console.

Prerequisites

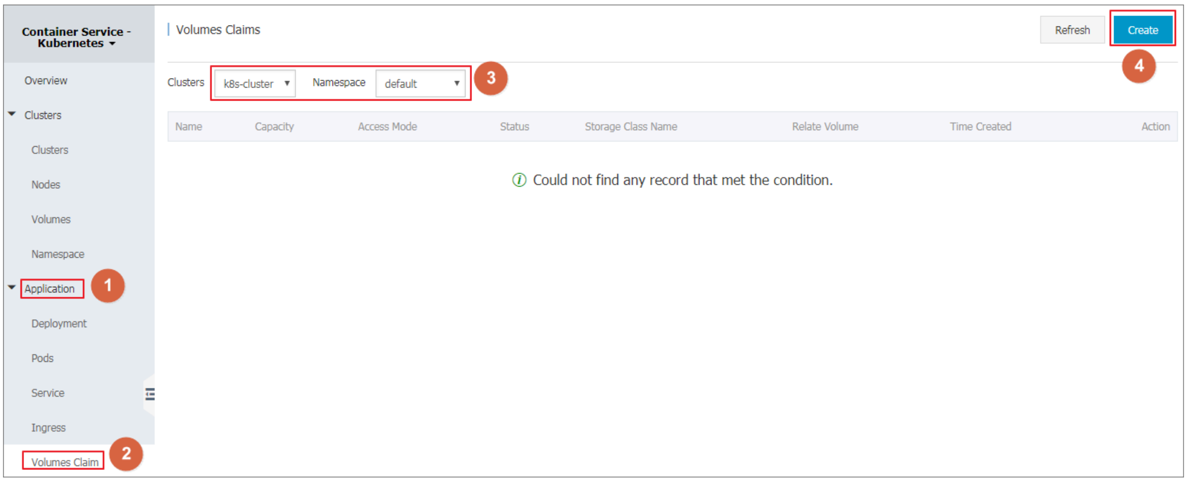
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a storage volume. In this example, use a cloud disk to create a cloud storage volume. For more information, see *Use Alibaba Cloud cloud disk volumes*.

  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.

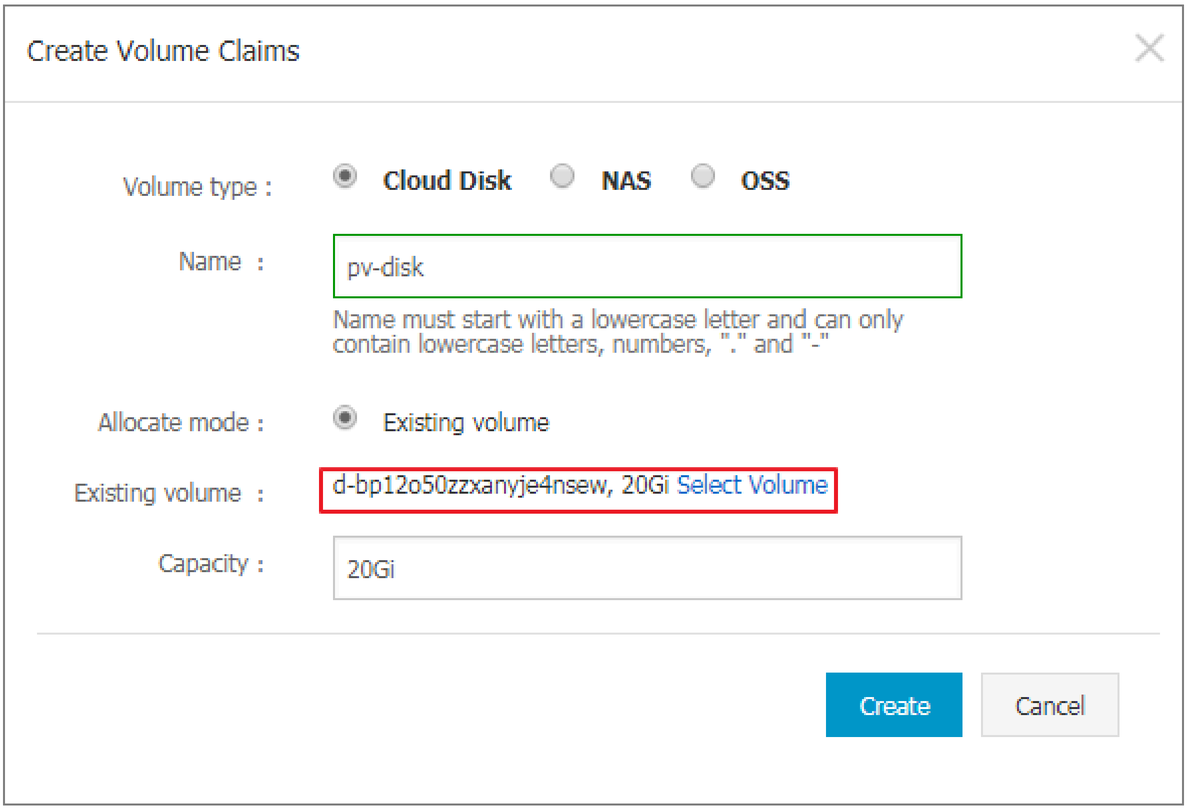
Context

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.



4. Complete the configurations in the Create Volume Claim dialog box, and click Create.

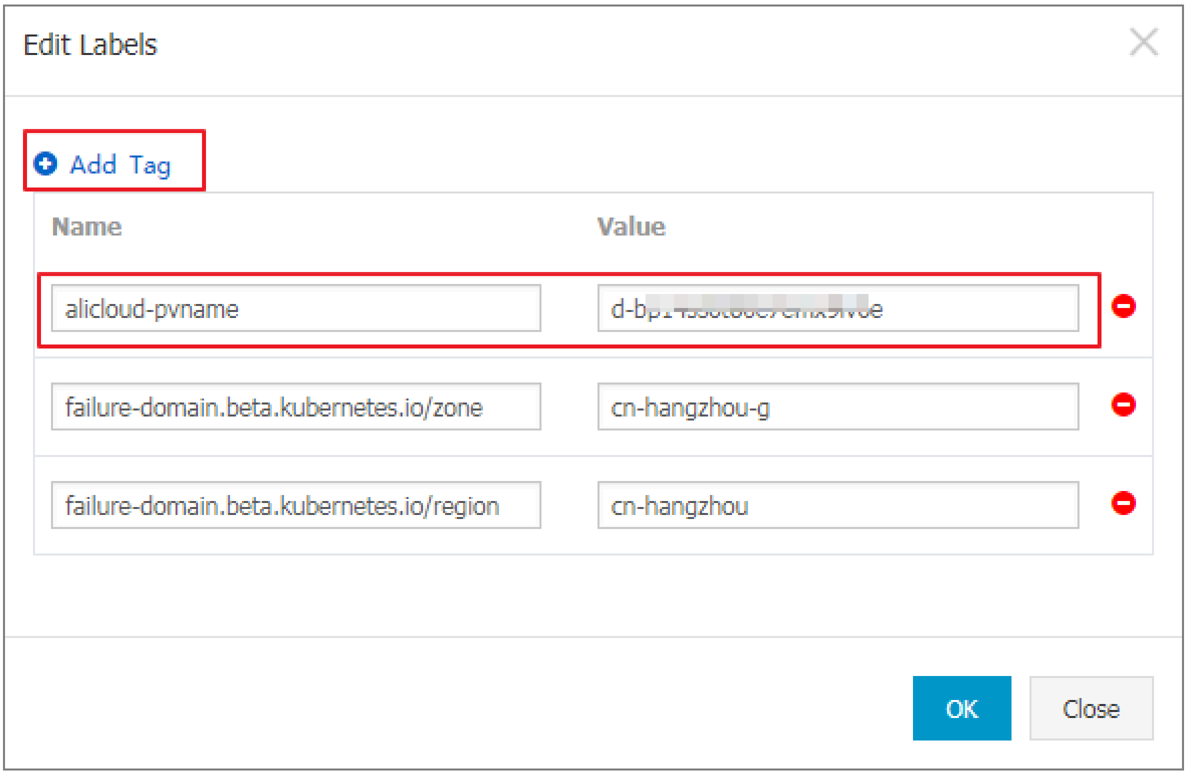


- Volume claim type: Consistent with storage volume, including cloud disk, NAS, and OSS types.
- Name: Enter the storage volume claim name.
- Distribution mode: Currently, only existing storage volumes are supported.
- Existing storage volume: Select to bind the storage volume of this type.
- Total: 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.



5. Return to the Volumes Claims list, you can see that the newly created storage claim appears in the list.

1.11.7 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_149*.

Prerequisites

- You have created a Kubernetes cluster. For more information, see *#unique_40*.
- 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_150*.

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_96* 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.

1.12 Log management

1.12.1 Application log management

A Kubernetes cluster that runs on Alibaba Cloud Container Service provides you with multiple methods to manage application logs.

- Following the instructions of *Use Log Service to collect Kubernetes cluster logs*, you can make the best use of the functions provided by Alibaba Cloud Log Service, such as log statistics and analysis.
- With *Log-pilot*, an open source project provided by Alibaba Cloud Container Service, and *A solution to log collection problems of Kubernetes clusters by using log-pilot, Elasticsearch, and Kibana*, you can easily build your own application log clusters.

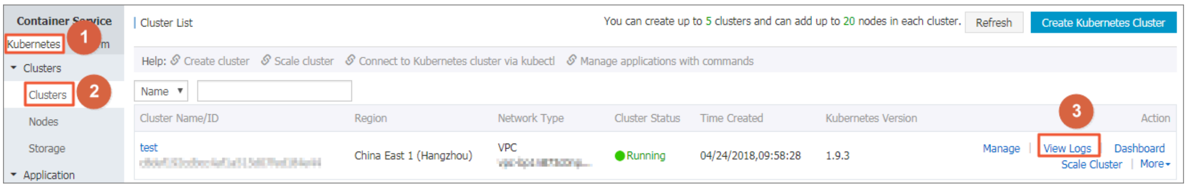
1.12.2 View cluster logs

Context

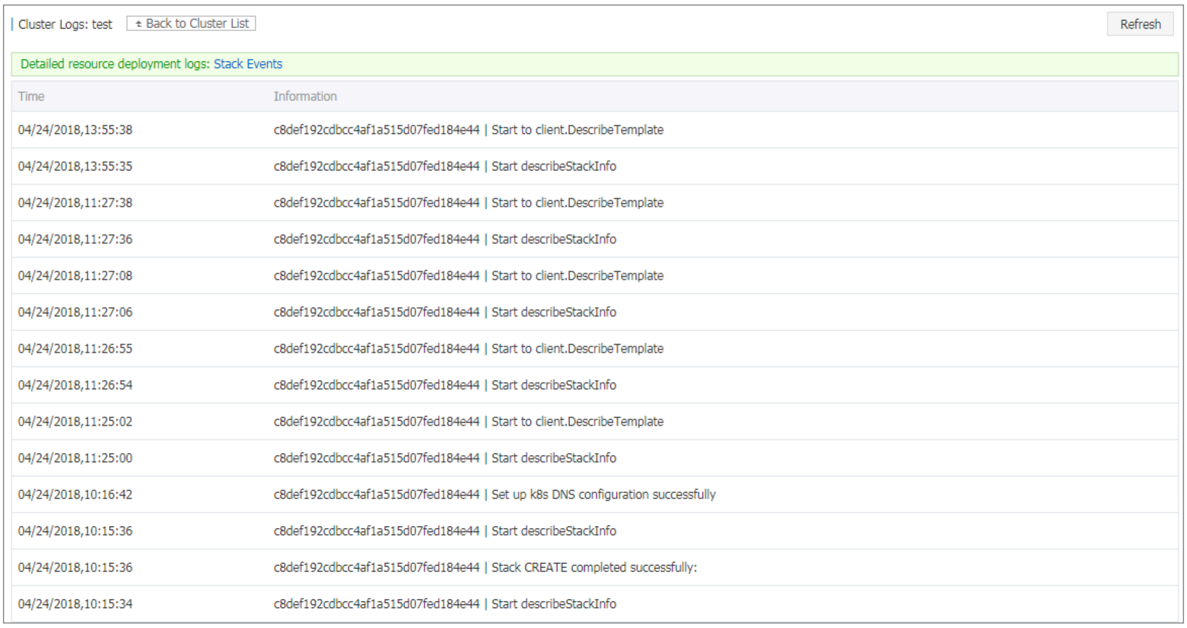
You can view the cluster operation logs by using the simple log service of Container Service.

Procedure

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



View the cluster operation information.



Cluster Logs: test Back to Cluster List Refresh

Detailed resource deployment logs: Stack Events

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

1.12.3 Use Log Service to collect Kubernetes cluster logs

Log Service is integrated with Kubernetes clusters of Alibaba Cloud Container Service. You can enable Log Service when creating a cluster to quickly collect container logs for the Kubernetes cluster, such as the standard output of the container and text files of the container.

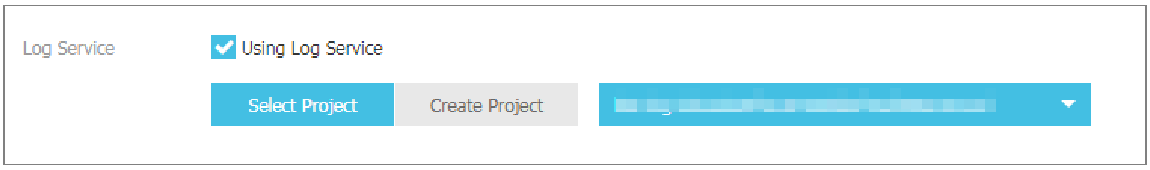
Enable Log Service when creating a Kubernetes cluster

If you have not created any Kubernetes clusters, follow steps in this section to enable Log Service:

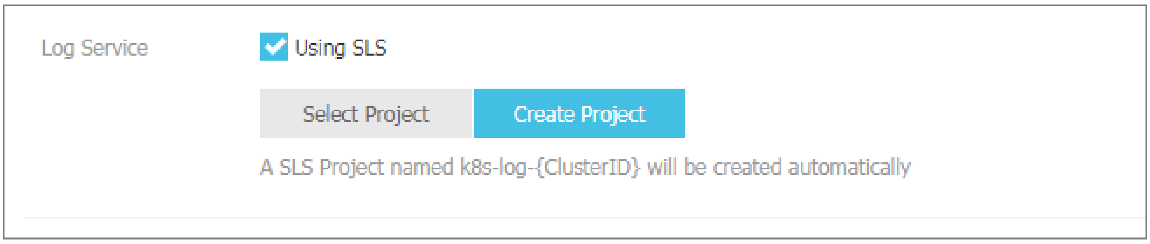
1. Log on to the *Container Service console*.
2. Click Clusters in the left-side navigation pane and click Create Kubernetes Cluster in the upper-right corner.

3. For how to configure a cluster on the creation page, see *Create a Kubernetes cluster*.
4. Drag to the bottom of the page and select the Using Log Service check box. The log plug-in will be installed in the newly created Kubernetes cluster.
5. When you select the Using Log Service check box, project options are displayed. A project is the unit in Log Service to manage logs. For more information about projects, see *Project* . Currently, two ways of using a project are available:

   - Select an existing project to manage collected logs.

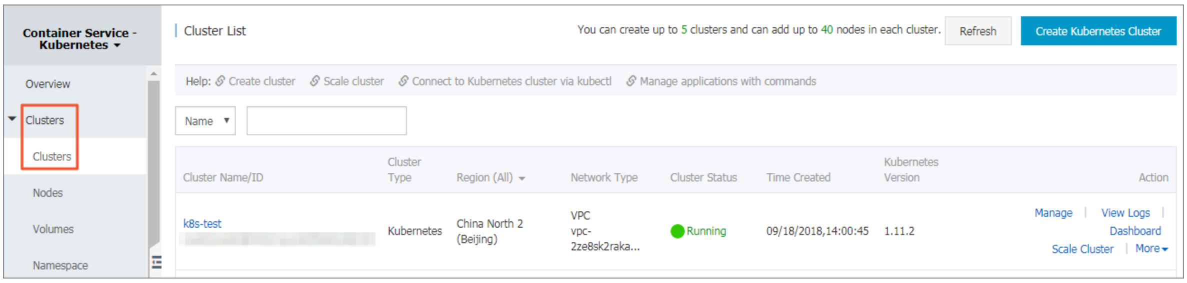
   

   - The system automatically creates a new project to manage collected logs. The project is automatically named `k8s-log-{ClusterID}`, where ClusterID represents the unique identifier of the created Kubernetes cluster.

6. After you complete the configurations, click Create in the upper-right corner. In the displayed dialog box, click OK.

   After the cluster creation is completed, the newly created Kubernetes cluster is displayed on the cluster list page.

Manually install Log Service components in a created Kubernetes cluster

If you have created a Kubernetes cluster, following instructions in this section to use Log Service:

- Log Service components are not installed. Manually install the components.

- Log Service components are installed but in an earlier version. Upgrade the components. If you do not upgrade the components, you can only use the Log Service console or custom resource definition (CRD) to configure log collection.

Check the Log Service component version

1. Configure the local kubeconfig to connect to the Kubernetes cluster through kubectl.

   For information about the configuration, see *Connect to a Kubernetes cluster by using kubectl*.
2. Run the following command to fast determine whether an upgrade or migration operation is required:

```
$ kubectl describe daemonsets -n kube-system logtail-ds | grep
ALICLOUD_LOG_DOCKER_ENV_CONFIG
```

   - If `ALICLOUD_LOG_DOCKER_ENV_CONFIG: true` is output, the components can be used directly without requiring upgrade or migration.
   - If other results are output, check the components further.
3. Run the following command to determine whether Helm is used to install the components.

```
$ helm get alibaba-log-controller | grep CHART
CHART: alibaba-cloud-log-0.1.1
```

   - 0.1.1 in the output indicates the version of the Log Service components. Please use the version of 0.1.1 and later. If the version is too early, please see *Upgrade Log Service components* to upgrade the components. If you have used Helm to install the components of a valid version, you can skip next steps.
   - If no results are output, the components are installed by using Helm. But the DaemonSet installation method might be used. Follow the next step to check further.
4. DaemonSet can be an old one or a new one:

```
$ kubectl get daemonsets -n kube-system logtail
```

   - If no result is output or `No resources found.` is output, the Log Service components are not installed. For information about the installation method, see *Manually install Log Service components*.

- If the correct result is output, an old DaemonSet is used to install the components which require upgrade. For information about upgrading the components, see *Upgrade Log Service components*.

Manually install Log Service components

1. Configure the local kubeconfig to connect to the Kubernetes cluster through kubectl.

   For information about the configuration, see *Connect to a Kubernetes cluster by using kubectl*.

2. Replace a parameter and run the following command.

   Replace ${your_k8s_cluster_id} in the following command with your Kubernetes cluster ID, and then run the command.

   ```
   wget https://acs-logging.oss-cn-hangzhou.aliyuncs.com/alicloud-k8s
   -log-installer.sh -O alicloud-k8s-log-installer.sh; chmod 744 ./
   alicloud-k8s-log-installer.sh; ./alicloud-k8s-log-installer.sh --
   cluster-id ${your_k8s_cluster_id} --ali-uid ${your_ali_uid} --region
   -id ${your_k8s_cluster_region_id}
   ```

   Parameter descriptions:

   - your_k8s_cluster_id: You Kubernetes cluster ID.
   - your_ali_uid: You account ID of Alibaba Cloud, which can be viewed in the user info.
   - your_k8s_cluster_region_id: The region in which you Kubernetes cluster resides, which can be found in*Regions and zones*. For example, if the cluster resides in Hangzhou, the value of this parameter cn-hangzhou.

Installation example

```
[root@iZbp******biaZ ~]# wget https://acs-logging.oss-cn-hangzhou
.aliyuncs.com/alicloud-k8s-log-installer.sh -O alicloud-k8s-log-
installer.sh; chmod 744 ./alicloud-k8s-log-installer.sh; ./alicloud-
k8s-log-installer.sh --cluster-id c77a****************0106 --ali-uid
19*********19 --region-id cn-hangzhou
--2018-09-28 15:25:33--  https://acs-logging.oss-cn-hangzhou.aliyuncs.
com/alicloud-k8s-log-installer.sh
Resolving acs-logging.oss-cn-hangzhou.aliyuncs.com... 118.31.219.217,
118.31.219.206
Connecting to acs-logging.oss-cn-hangzhou.aliyuncs.com|118.31.219.217
|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2273 (2.2K) [text/x-sh]
Saving to: ‘alicloud-k8s-log-installer.sh’
```

```
alicloud-k8s-log-installer.sh                    100
%[=========================================================================
   2.22K  --.-KB/s    in 0s

2018-09-28 15:25:33 (13.5 MB/s) - ‘alicloud-k8s-log-installer.sh’
saved [2273/2273]

--2018-09-28 15:25:33--  http://logtail-release-cn-hangzhou.oss-cn-
hangzhou.aliyuncs.com/kubernetes/alibaba-cloud-log.tgz
Resolving logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com...
118.31.219.49
Connecting to logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com
|118.31.219.49|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2754 (2.7K) [application/x-gzip]
Saving to: ‘alibaba-cloud-log.tgz’

alibaba-cloud-log.tgz                            100
%[=========================================================================
   2.69K  --.-KB/s    in 0s

2018-09-28 15:25:34 (79.6 MB/s) - ‘alibaba-cloud-log.tgz’ saved [2754/
2754]

[INFO] your k8s is using project : k8s-log-c77a92ec5a3ce4e64a1b
f13bde1820106
NAME:   alibaba-log-controller
LAST DEPLOYED: Fri Sep 28 15:25:34 2018
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1beta1/CustomResourceDefinition
NAME                                AGE
aliyunlogconfigs.log.alibabacloud.com  0s

==> v1beta1/ClusterRole
alibaba-log-controller  0s

==> v1beta1/ClusterRoleBinding
NAME                    AGE
alibaba-log-controller  0s

==> v1beta1/DaemonSet
NAME        DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE
SELECTOR  AGE
logtail-ds  2        2        0      2           0          <none>
     0s

==> v1beta1/Deployment
NAME                    DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
alibaba-log-controller  1        1        1           0          0s

==> v1/Pod(related)
NAME                                     READY  STATUS
RESTARTS  AGE
logtail-ds-6v979                         0/1    ContainerCreating  0
       0s
logtail-ds-7ccqv                         0/1    ContainerCreating  0
       0s
alibaba-log-controller-84d8b6b8cf-nkrkx  0/1    ContainerCreating  0
       0s

==> v1/ServiceAccount
```

```
NAME                     SECRETS   AGE
alibaba-log-controller   1         0s


[SUCCESS] install helm package : alibaba-log-controller success.
```

Upgrade Log Service components

If you have installed Log Service components of an early version through Helm or DaemonSet, upgrade or migrate the components as follows.

> **Note:**
> To perform the following operations, first log on to the master node of your Kubernetes cluster of Alibaba Cloud Container Service. For information about the logon method, see *Connect to a Kubernetes cluster by using kubectl*.

Use Helm to upgrade Log Service components (recommended)

1. Run the following command to download the latest Helm package of Log Service components:

```
wget http://logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com
/kubernetes/alibaba-cloud-log.tgz -O alibaba-cloud-log.tgz
```

2. Upgrade the components by using helm upgrade. The command is as follows:

```
helm get values alibaba-log-controller --all > values.yaml && helm
upgrade alibaba-log-controller alibaba-cloud-log.tgz --recreate-pods
 -f values.yaml
```

Use DaemonSet to upgrade Log Service components

You can upgrade Log Service components by modifying the DaemonSet template. If your image account is acs, upgrade the image tag to the latest version that can be viewed in *Container Registry*. If your image account is acs, upgrade the image tag to the latest version that can be viewed in *Container Registry*.

> **Note:**
> - If upgrading the tag has not enabled a rolling update of Logtail, you must manually remove the Logtail pod to trigger a Logtail update.
> - You need to check whether Logtail runs on all nodes, including Master nodes. If Logtail does not run on all nodes, you must set *tolerations* for Logtail.

```
tolerations:
```

```
- operator: "Exists"
```

For more information, see *Latest Helm package configurations*.

DaemonSet migrate

This upgrade method is applicable to the situation that you find the components are installed through the old DaemonSet when you check the Log Service component version. This method does not support configuring Log Service in Container Service. You can upgrade the components as follows:

1. At the end of the installation command, add a parameter which is the name of the project of Log Service used by your Kubernetes cluster.

   For example, if the project name is k8s-log-demo and the cluster ID is c12ba2028cxxxxxxxxxx6939f0b, then the installation command is:

   ```
   wget https://acs-logging.oss-cn-hangzhou.aliyuncs.com/alicloud-k8s
   -log-installer.sh -O alicloud-k8s-log-installer.sh; chmod 744 ./
   alicloud-k8s-log-installer.sh; ./alicloud-k8s-log-installer.sh --
   cluster-id c12ba2028cxxxxxxxxxx6939f0b --ali-uid 19*********19 --
   region-id cn-hangzhou --log-project k8s-log-demo
   ```
2. After you complete the installation, log on the *Log Service console*.
3. After you complete the installation, log on the *Log Service console*.
4. In the Log service console, apply the history collection configuration of the project and Logstore to the new machine group `k8s-group-${your_k8s_cluster_id}`.
5. After one minute, the history collection configuration is unbound from the history machine group.
6. When log collection is normal, you can delete the previously installed Logtail DaemonSet.

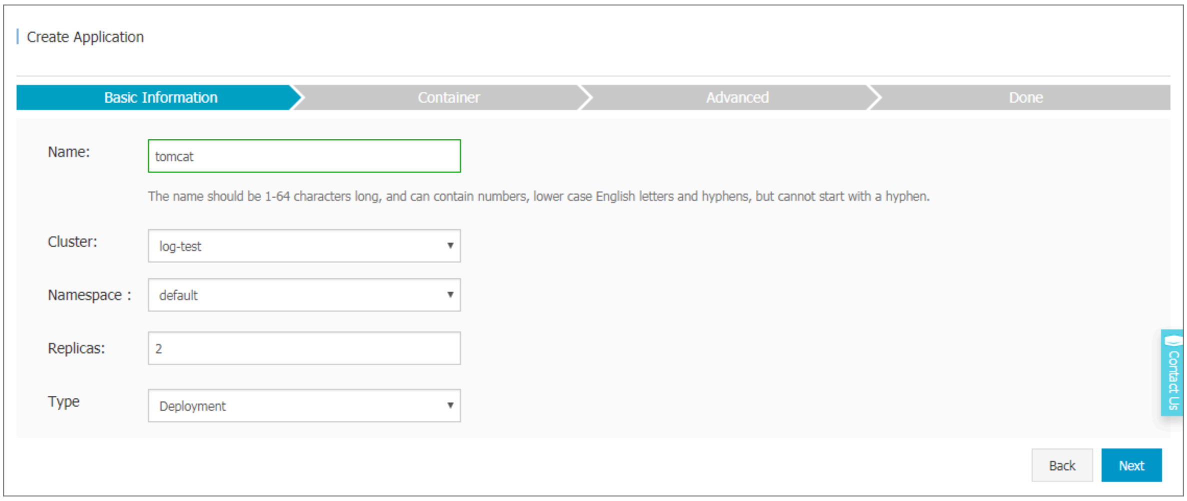
> Note:
> During the upgrade, some logs are duplicated. The CRD configuration management takes effect only for the configuration created by using CRD. The history configuration does not support the CRD management because the history configuration is created by using the non-CRD mode.

Configure Log Service when creating an application

Container Service allows you to configure Log Service to collect container logs when creating an application. Currently, you can use the console or a YAML template to create an application.

Create an application by using the console

1. Log on to the *Container Service console*.
2. 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, and then click Next.



4. On the Container page, select the Tomcat image and configure container log collection.

   The following describes only configurations related to Log Service. For information about other application configurations, see *Create a deployment application by using an image*.
5. Configure Log Service. Click the + sign to create a configuration which consists of a Logstore name and a log path.

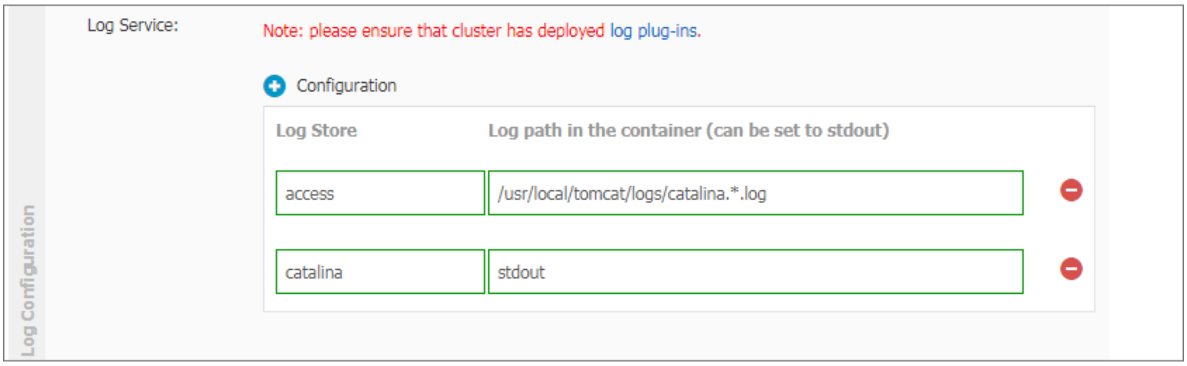
   - Logstore name: Specify a Logstore in which collected logs are stored. If your specified Logstore does not exist, the system automatically creates the Logstore in the project of Log Service with which the cluster is associated .

     Note:
     A Logstore name cannot contain underscores (_). You can use hyphens (-) instead.
   - Log path: Specify the path where logs to be collected reside. For example, use `/usr/local/tomcat/logs/catalina. *.log` to collect text logs of tomcat.

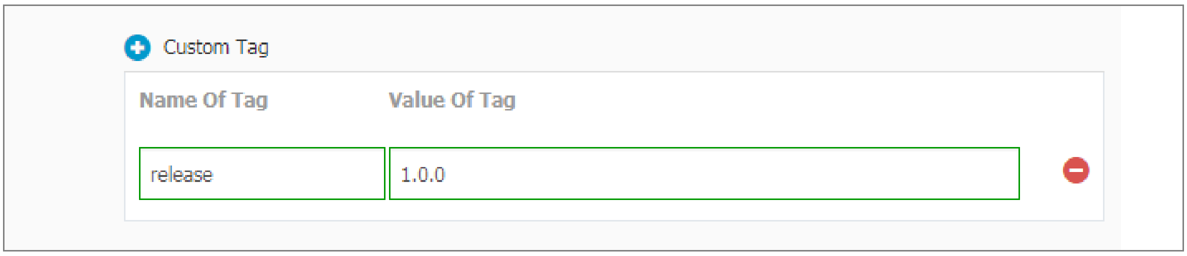
     Note:

If you specify the log path as stdout, the container standard output and standard error output will be collected.

Each configuration is automatically created as a configuration for the corresponding Logstore. By default, the simple mode (by row) is used to collect logs. To use more collection modes, log on go to the Log Service console, and enter the corresponding project (prefixed with k8s-log by default) and Logstore to modify the configuration.



6. Custom tag. Click the + sign to create a new custom tag. Each custom tag is a key-value pair which will be added to collected logs. You can use a custom tag to mark container logs. For example, you can create a custom tag as a version number.



7. When you complete all the configurations of the container, click Next in the upper-right corner to perform further configurations. For more information, see *Create a deployment application by using an image*.

Create an application by using a YAML template

1. Log on to the *Container Service console*.
2. Under the Kubernetes menu, click Application > Deployment in the left-side navigation pane, and then click Create by Template in the upper-right corner.
3. The syntax of the YAML template is the same as the Kubernetes syntax. To specify the collection configuration for the container, you need to use env to add collection

configuration and custom tag for the container, and create corresponding volumeMounts and volumns. The following is a simple pod example:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-demo
spec:
  containers:
  - name: my-demo-app
    image: 'registry.cn-hangzhou.aliyuncs.com/log-service/docker-log
-test:latest'
    env:
    ######### Configure environment variables  ###########
    - name: aliyun_logs_log-stdout
      value: stdout
    - name: aliyun_logs_log-varlog
      value: /var/log/*.log
    - name: aliyun_logs_mytag1_tags
      value: tag1=v1
    ###############################
    ######### Configure vulume mount ###########
    volumeMounts:
    - name: volumn-sls-mydemo
      mountPath: /var/log
  volumes:
  - name: volumn-sls-mydemo
    emptyDir: {}
  ###############################
```

- Configure three parts in order based on your needs.
- In the first part, use environment variables to create your collection configurat ion and custom tag. All environment variables related to configuration are prefixed with `aliyun_logs_`.
- Rules for creating the collection configuration are as follows:

  ```
  - name: aliyun_logs_{Logstore name}
    value: {log path}
  ```

  In the example, create two collection configurations. The `aliyun_logs_log-stdout` env creates a configuration that contains a Logstore named log-stdout and the log path of stdout. The standard output of the container is collected and stored to the Logstore named log-stdout.

  Note:

  A Logstore name cannot contain underscores (_). You can use hyphens (-) instead.
- Rules for creating a custom tag are as follows:

  ```
  - name: aliyun_logs_{a name without '_'}_tags
  ```

```
value: {Tag name}={Tag value}
```

After a tag is configured, when logs of the container are collected, fields corresponding to the tag are automatically attached to Log Service.

- If you specify a non-stdout log path in your collection configuration, create corresponding volumnMounts in this part.

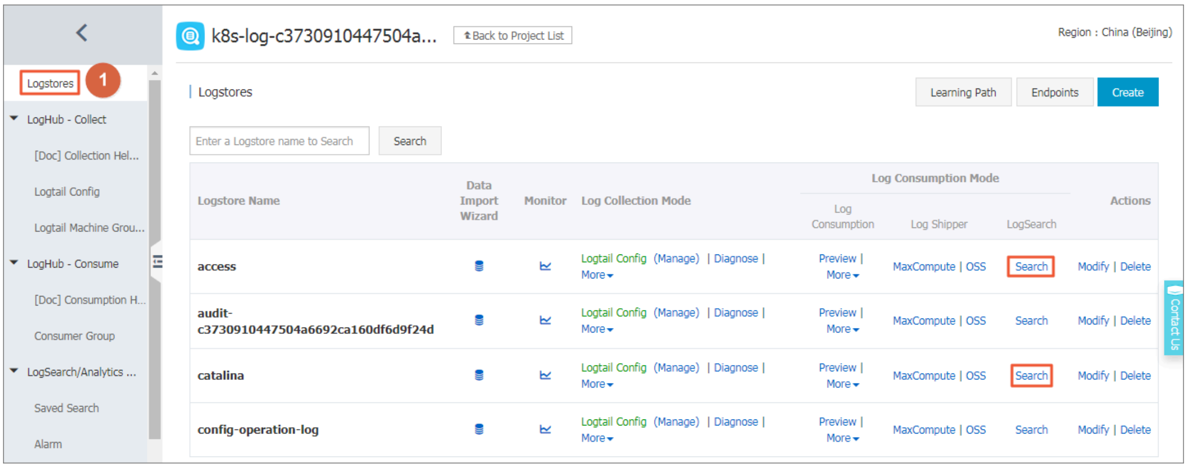
  In the example, the `/var/log/*.log` log path is added to the collection configuration, therefore, the `/var/log` volumeMounts is added.

4. When you complete a YAML template, click DEPLOY to deliver the configurations in the template to the Kubernetes cluster to execute.

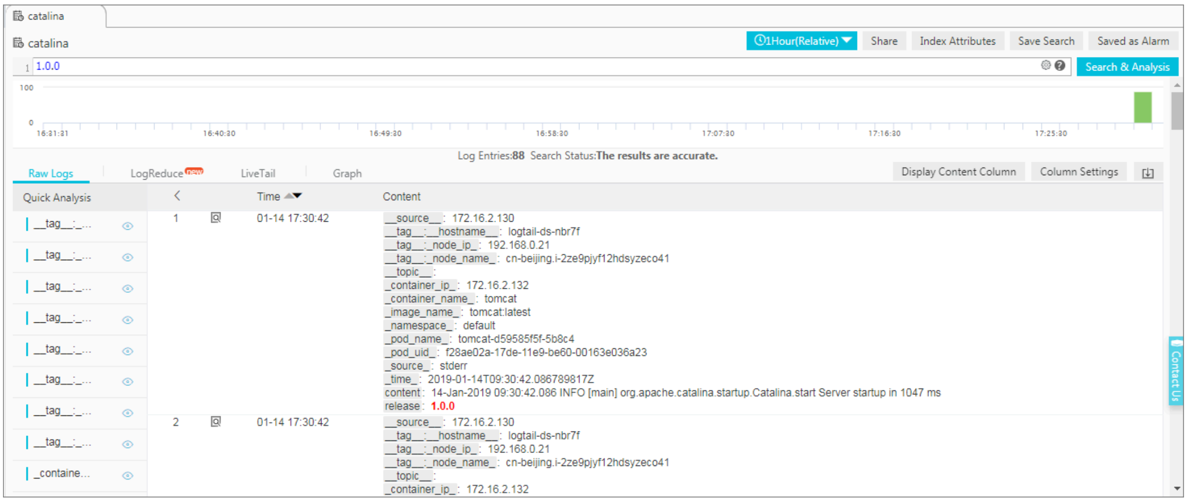
View logs

In this example, view logs of the tomcat application created in the console. After you complete the application configuration, logs of the tomcat application are collected and stored to Log Service. You can view your logs as follows:

1. Log on to the *Log Service console*.
2. Log on to the *Log Service console*.
3. In the console, select the project (k8s-log-{Kubernetes cluster ID} by default) corresponding to the Kubernetes cluster.
4. In the Logstore list, locate the Logstore specified in your configuration and click Search.



5. In this example, on the log search page, you can view the standard output logs of the tomcat application and text logs in the container, and you can find your custom tag is attached to log fields.



More information

1. By default, the system use the simple mode to collect your data, that is, to collect data by row without parsing. To perform more complex configurations, see the following Log Service documents and log on to the Log Service console to modify configurations.
    - *Container text logs*
    - *Container stdout*
2. Currently, Log Service uses plug-ins to collect the standard output logs of containers. You can configure more plug-ins to process collected logs further, such as to filter and extract fields.
3. In addition to configuring log collection through the console, you can also directly collect logs of the Kubernetes cluster through the CRD configuration. For more information, see *Configure Kubernetes log collection on CRD*.
4. For troubleshooting exceptions, see *Troubleshoot collection errors*.

1.12.4 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_40* or *#unique_163*.
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_164*. 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.

1.12.5 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.

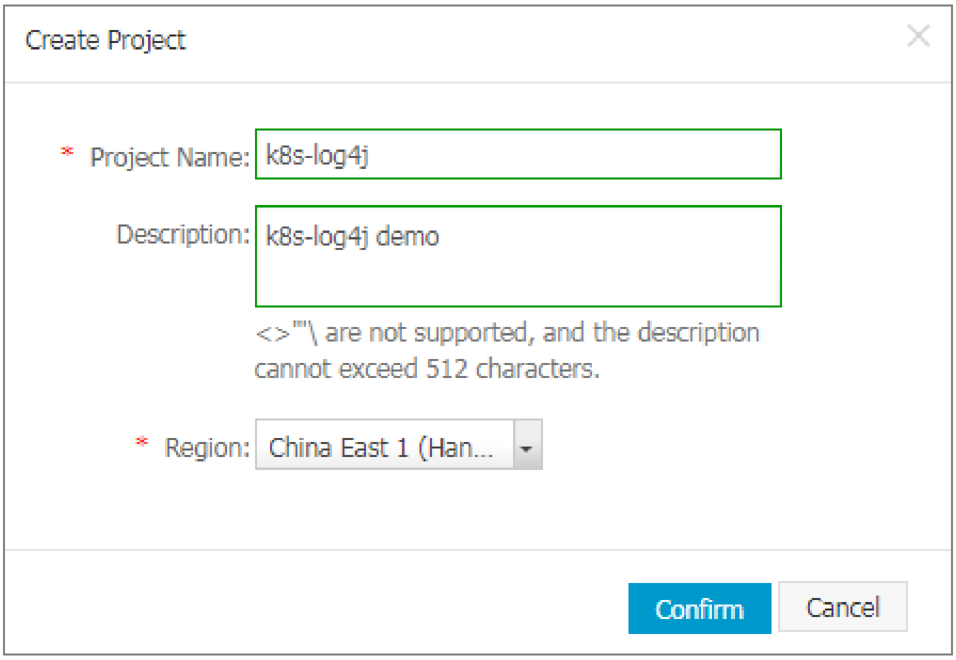
  In this example, create a Kubernetes cluster in the region of China East 1 (Hangzhou).
- Enable AccessKey or Resource Access Management (RAM). Make sure you have sufficient access permissions. Use the AccessKey in this example.

Step 1 Configure Log4jAppender in Alibaba Cloud Log Service

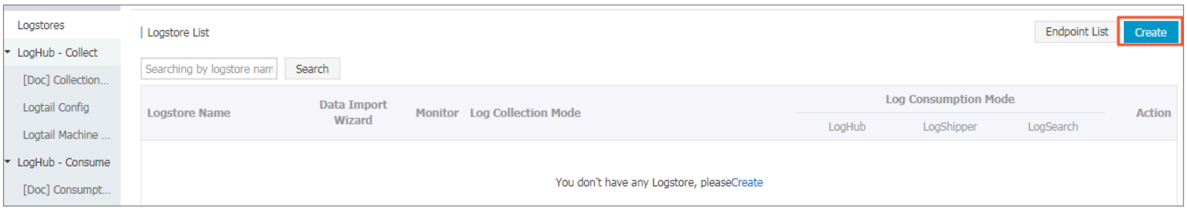
1. Log on to the *Log Service console*.
2. On the Project List page, click Create Project in the upper-right corner. Complete the configurations and then click Confirm to create the project.

   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.

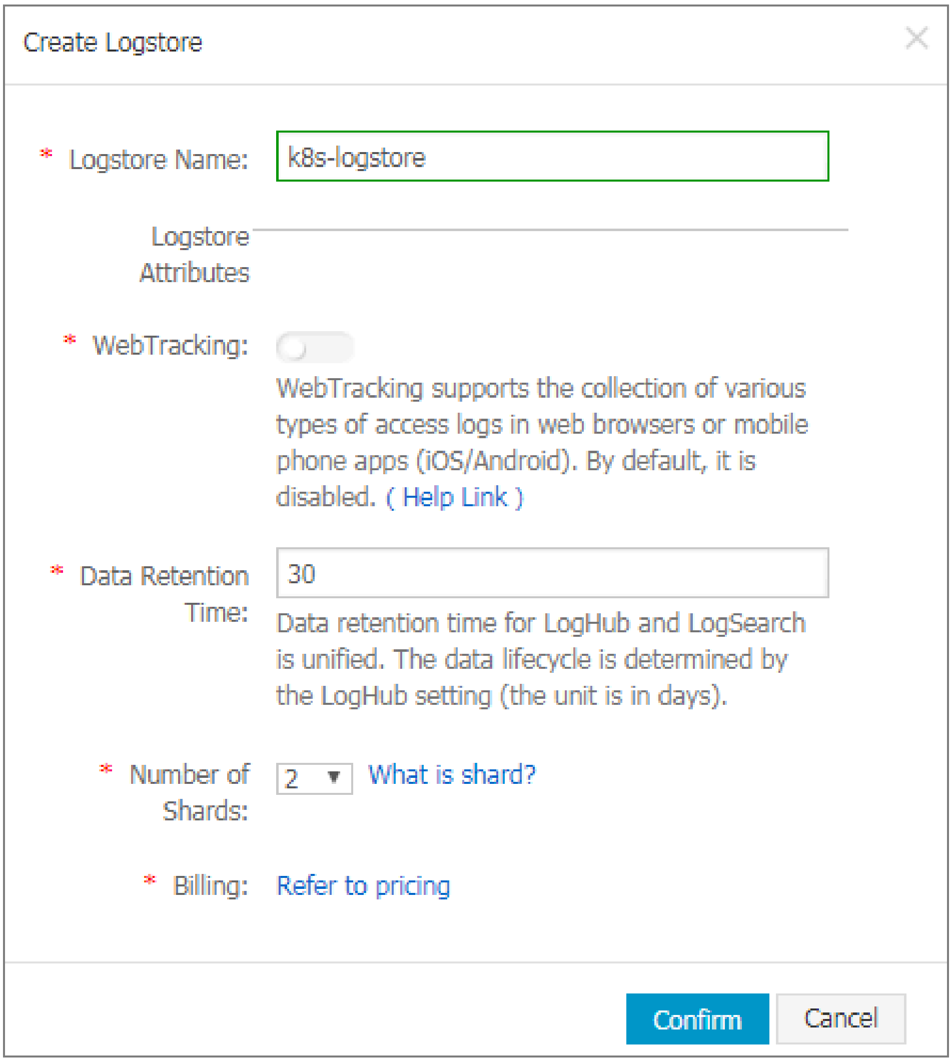


3. After being created, the project k8s-log4j is displayed on the Project List page. Click the project name.
4. The Logstore List page appears. Click Create in the upper-right corner.

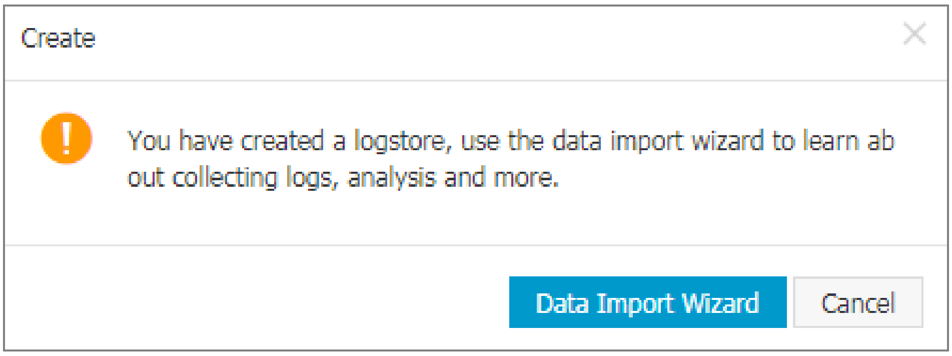


5. Complete the configurations and then click Confirm.

   In this example, create a Logstore named k8s-logstore.

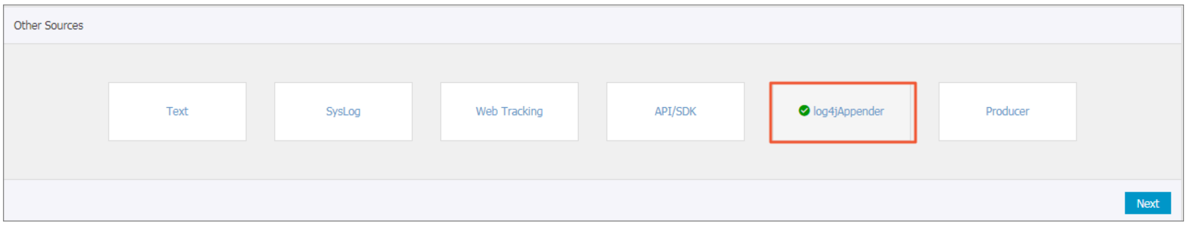


6. Then, a dialog box asking you to use the data import wizard appears.



7. Click Data Import Wizard. In the Select Data Source step, select log4jAppender under Other Sources and then complete the configurations as instructed on the page.

   Use the default configurations in this example. Configure the settings according to the specific scenarios of log data.



Step 2 Configure Log4jAppender in the Kubernetes cluster

In this example, use the sample YAML files *demo-deployment* and *demo-service* for demonstration.

1. Connect to your Kubernetes cluster.

   For more information, see *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:

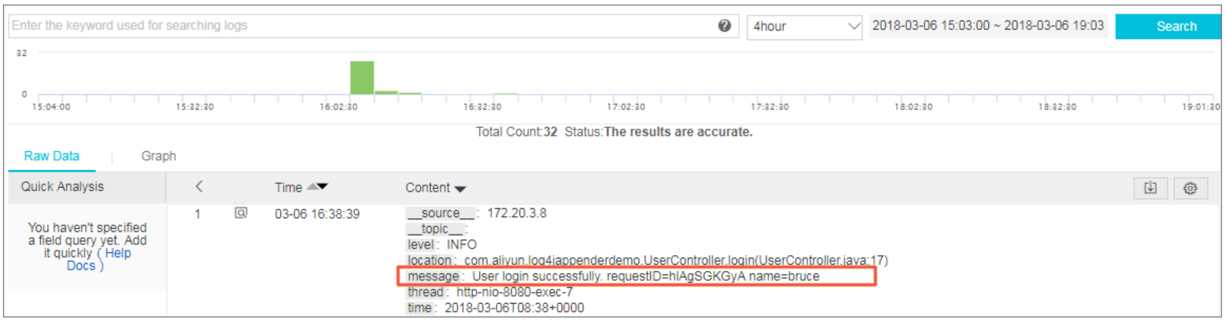
See *GitHub log4j-appender-demo* to view the complete collection of APIs.

```
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.



The output content of the log corresponds to the preceding command. This example demonstrates how to output the logs of the sample application to Alibaba Cloud Log Service. By completing the preceding steps, you can configure Log4JAppender in Alibaba Cloud and implement advanced functions such as collecting logs in real time , filtering data, and querying logs by using Alibaba Cloud Log Service.

1.13 Monitoring management

1.13.1 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, namely, the resource usage of a node or application. The monitoring system of Container Service for Kubernetes monitors node resource usage, cluster resource usage, and pod resource usage.
- Application, namely, internal metrics of an application. For example, The monitoring system collects statistics regarding the number of online users that use an application in real time, and performs service-level monitoring and alarming for the application by exposing ports.

The following are the objects monitored in a Kubernetes cluster:

- System components, which are built-in components of the Kubernetes cluster, such as apiserver, controller-manager, and etcd.
- Static resource entities, which include node resource status and kernel events.
- Dynamic resource entities, which are abstract workload entities of Kubernetes, such as deployment, DaemonSet, and pods.
- Customized application objects, which includes the data and metrics that require customization within an application.

To monitor system components and static resource entities, you need to specify monitoring methods for them in the configuration file.

To monitor dynamic resource entities, we recommend that you deploy the Prometheus monitoring system.

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:

> **Note:**

Some Prometheus components may fail to be deployed when you run this command for the first time because Prometheus components require a specific sequence to be deployed. If any exceptions occur during your first deployment, you need to run the command again.

```
cd prometheus-operator/contrib/kube-prometheus
kubectl apply -f manifests
```

3. Run the following command to set the access method for Prometheus:

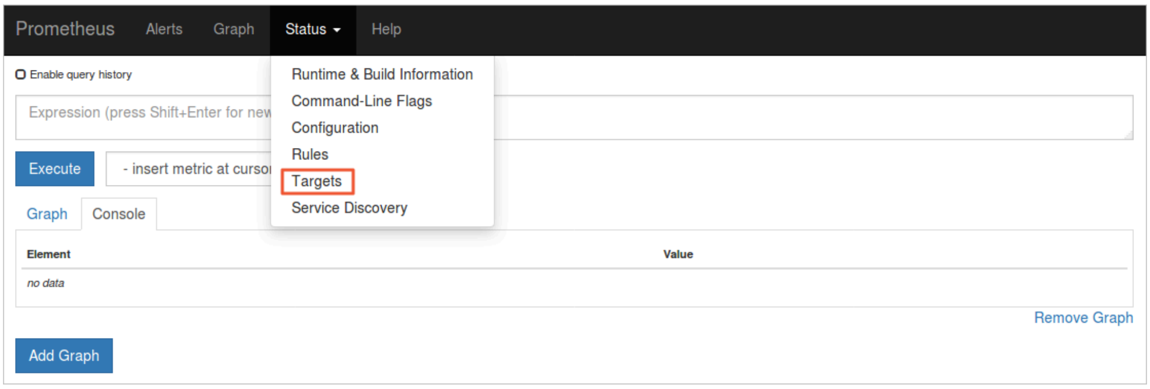
```
kubectl --namespace monitoring port-forward svc/prometheus-k8s 9090
```

4. View the deployment result

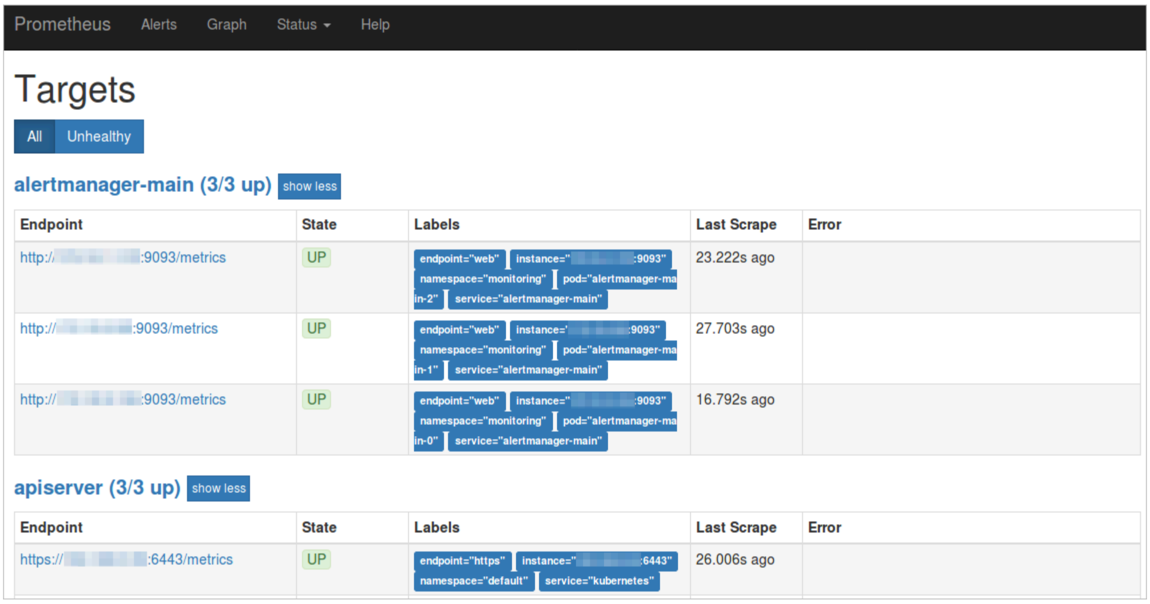
a. To view Prometheus, access`localhost:9090` in a browser.

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.



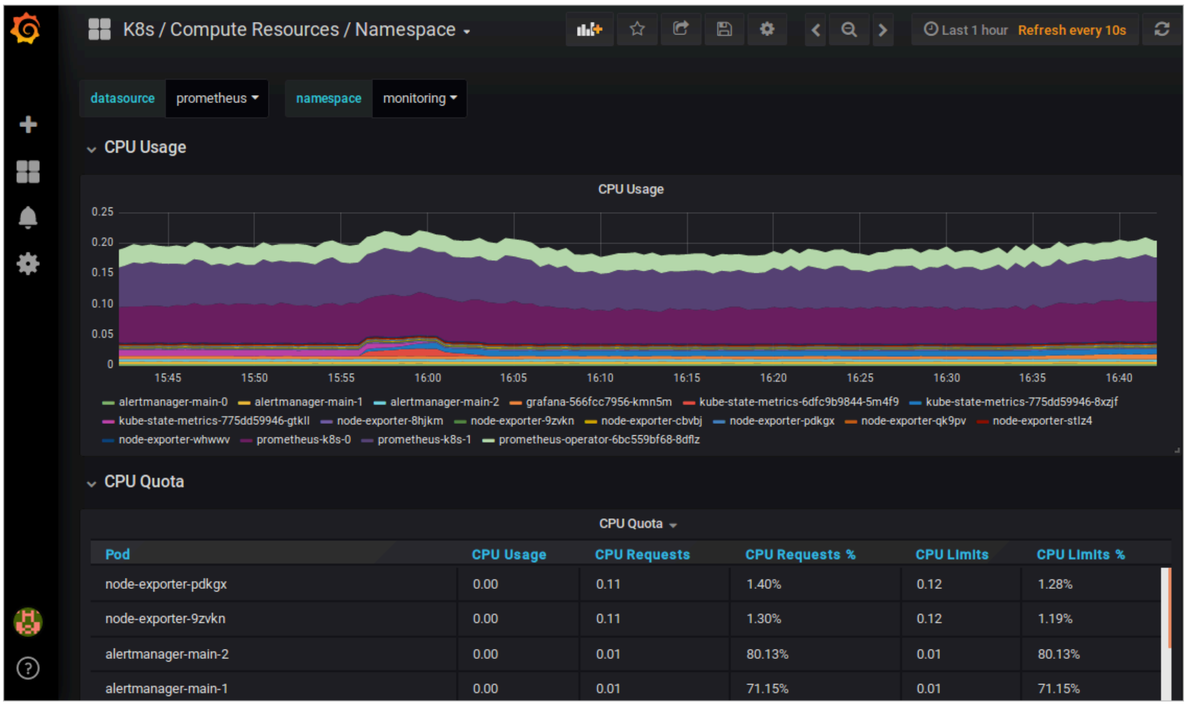
View and display data aggregation

1. Run the following command to access Grafana:

```
kubectl --namespace monitoring port-forward svc/grafana 3000
```

2. Access `localhost:3000` in your browser and then select a dashboard to view data aggregation.

> **Note:**
> The default user name and password are both admin.

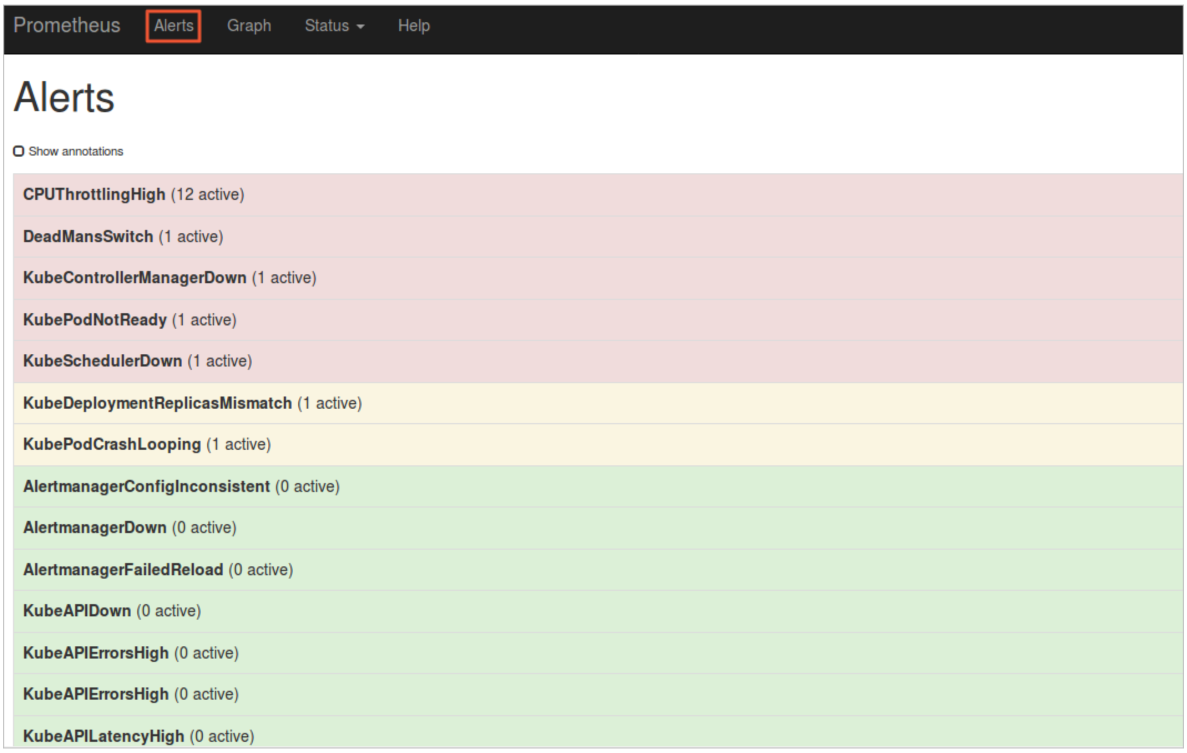


View alerting rules and set alert silencing

- View alerting rules

  Access `localhost:9090` in your browser and click the Alerts menu to view the current alerting rules.

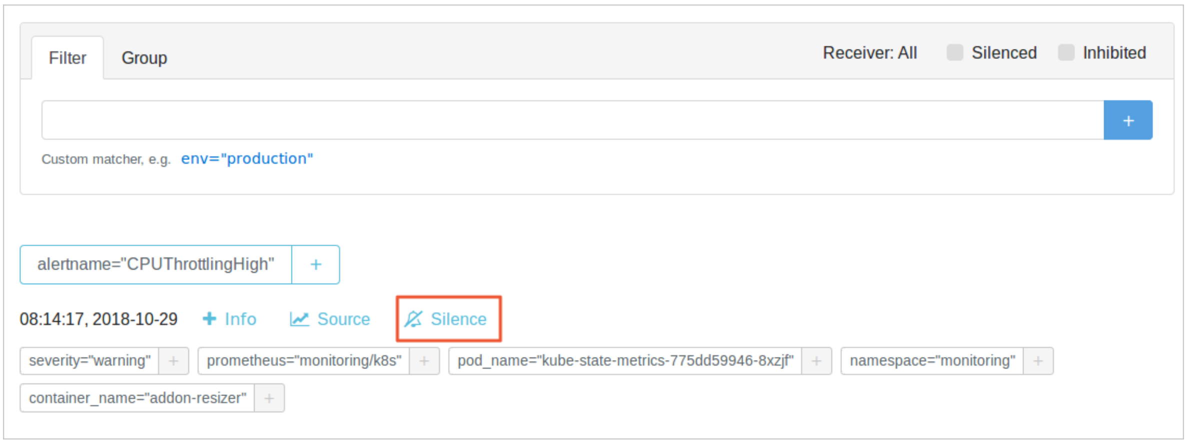
  - Red: indicates that an alert is triggered.
  - Green: indicates the normal status.



- Set alert silencing

Run the following command, open`localhost:9093` in your browser, and select Silenced to set alert silencing:

```
kubectl --namespace monitoring port-forward svc/alertmanager-main
9093
```



1.13.2 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

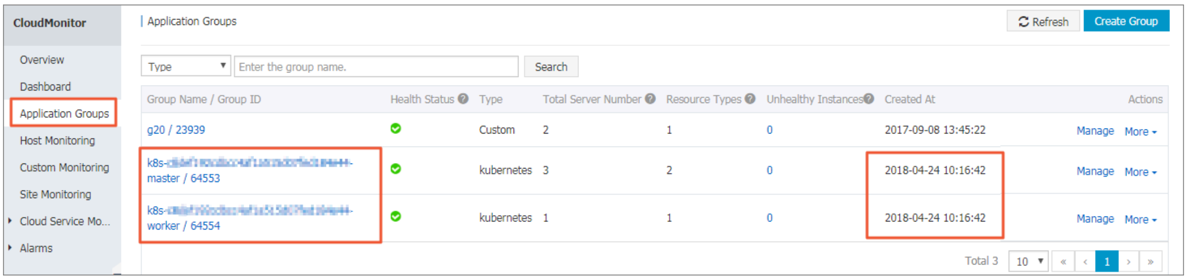
In the Operation & Maintenance (O&M) of IT infrastructure, monitoring and alarms facilitate daily O&M, system monitoring, troubleshooting, and debugging, and guarantee the reliability and security of O&M.

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 correspond ing 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.

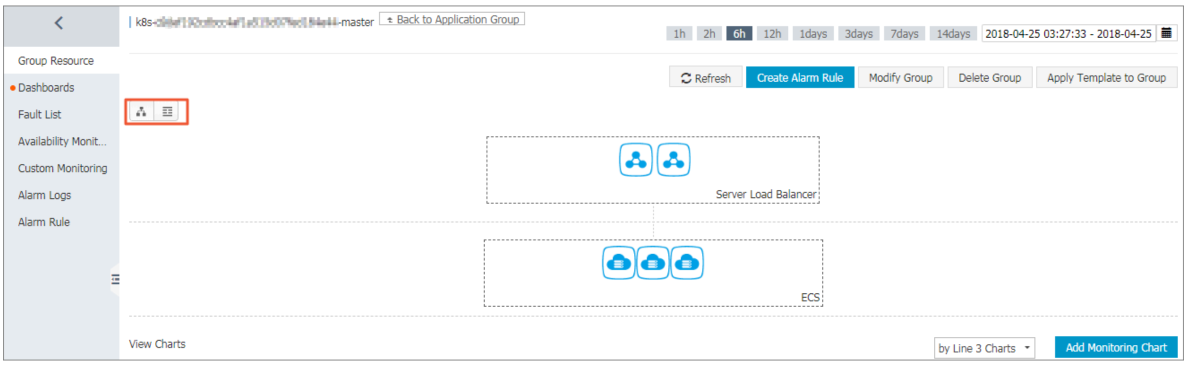
Procedure

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.

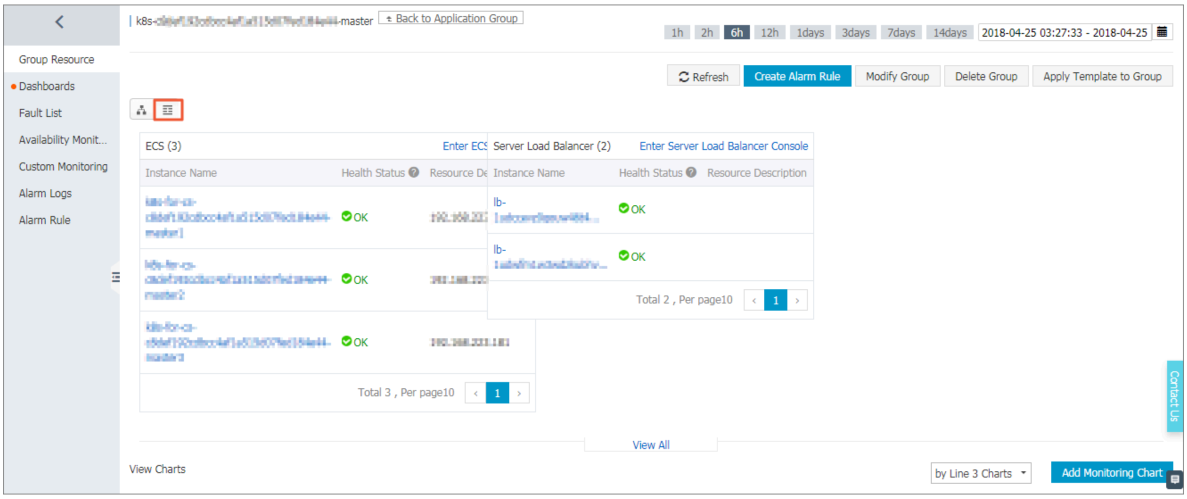


3. Click the Group Name to go to the group details page. You can view the resources contained in the group. For example, in a Master group of Kubernetes, you can see such resources as Elastic Compute Service (ECS) instances and Server Load Balancer (SLB) instances.

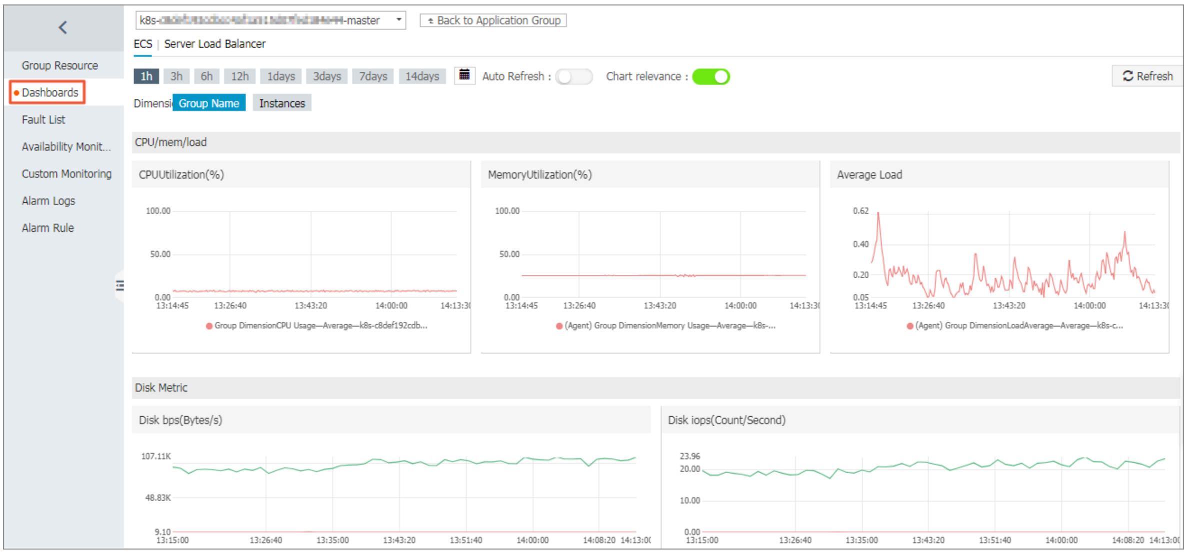
   Kubernetes has two types of nodes: Worker nodes and Master nodes. Master nodes generally contain management and control applications and the resources are required to be highly robust. Worker nodes are generally responsible for scheduling pods and the overall requirement on the resources focuses on scheduling 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.

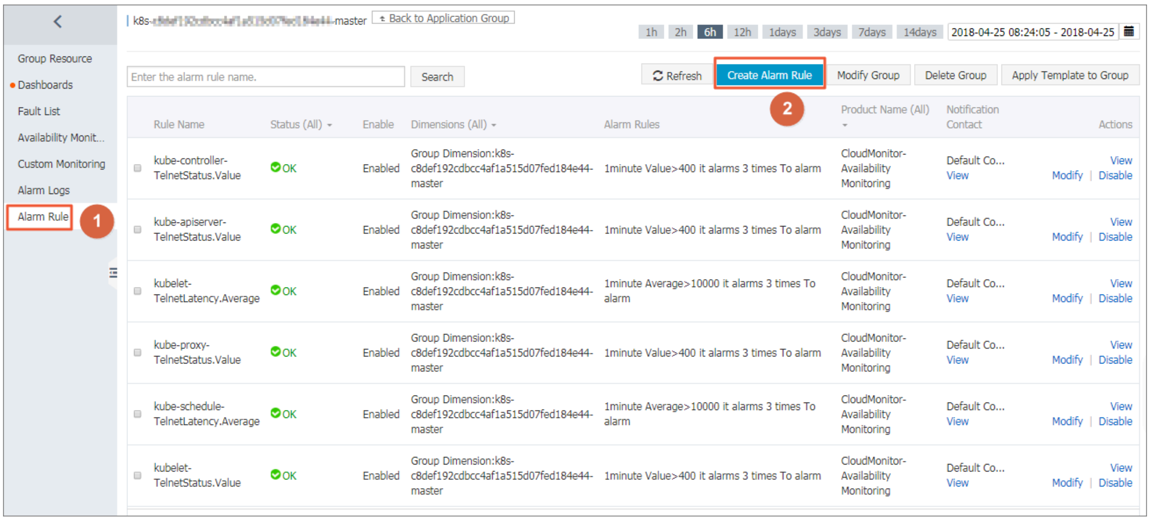


5. In the left-side navigation pane, click Dashboards to view the detailed monitoring metrics of each cloud product in the group.



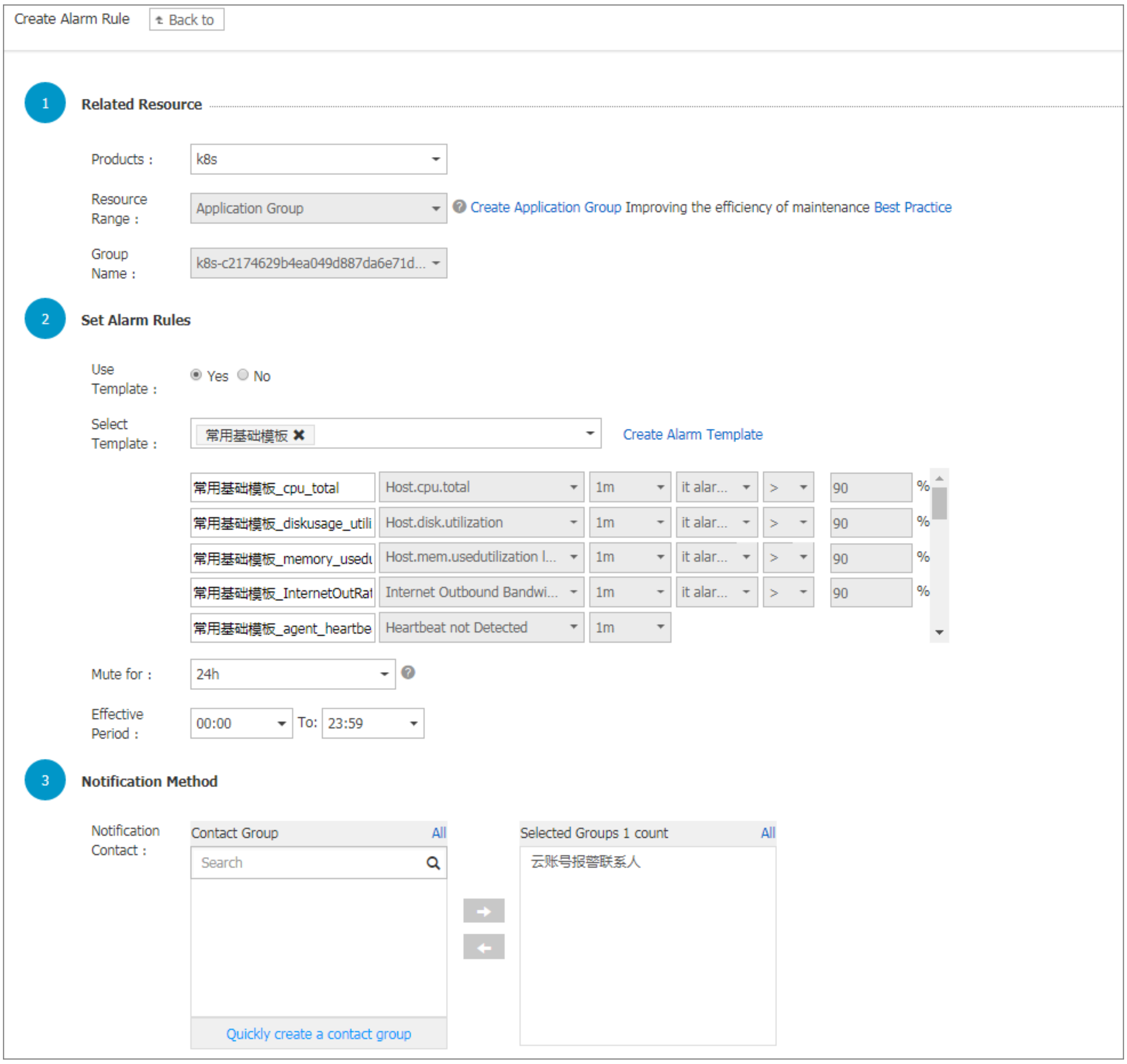
6. In the left-side navigation pane, click Alarm Rule. A list of existing alarm rules in the group is displayed. By default, the health of the core components of all nodes in the Mater group is checked.

   a. Click Create Alarm Rule to create an alarm rule for the group according to your business requirements.

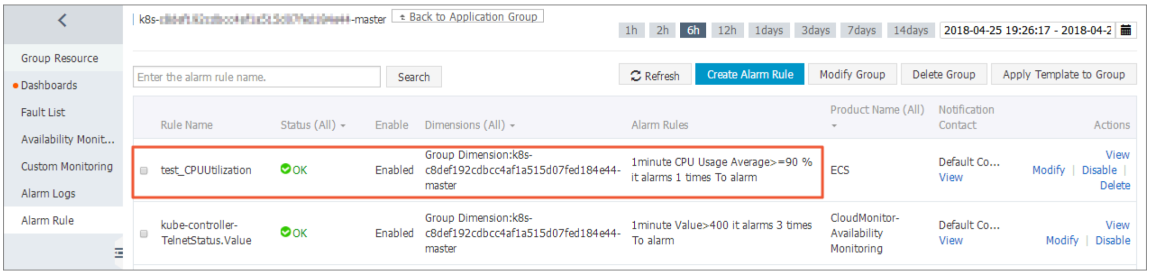


b. On the displayed Create Alarm Rule page, set the alarm rules.

- 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.



c. Click Confirm. The created alarm rule is displayed on the Alarm Rule page.



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.

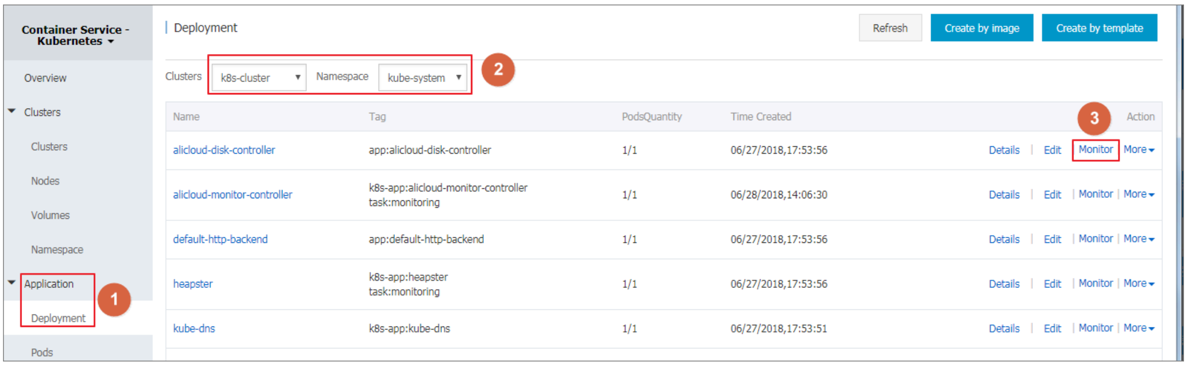
1.13.3 Integration and usage with CloudMonitor

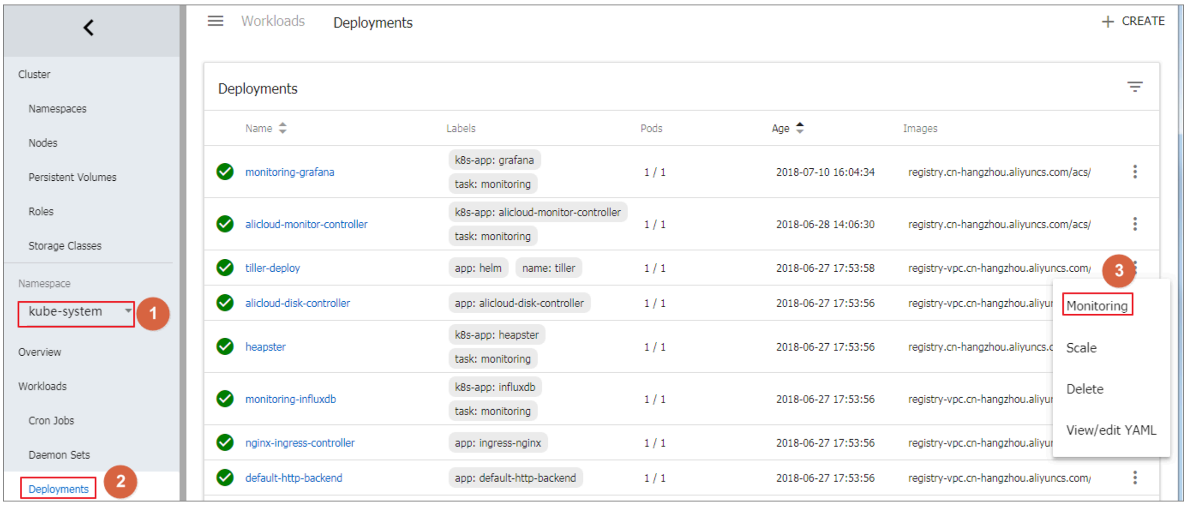
Prerequisites

Check whether `alicloud-monitor-controller` has been deployed in the `kube-system` namespace. If not, upgrade the version of the cluster.

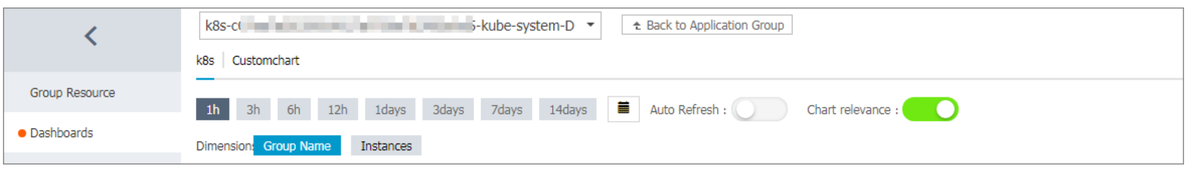
Procedure

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.

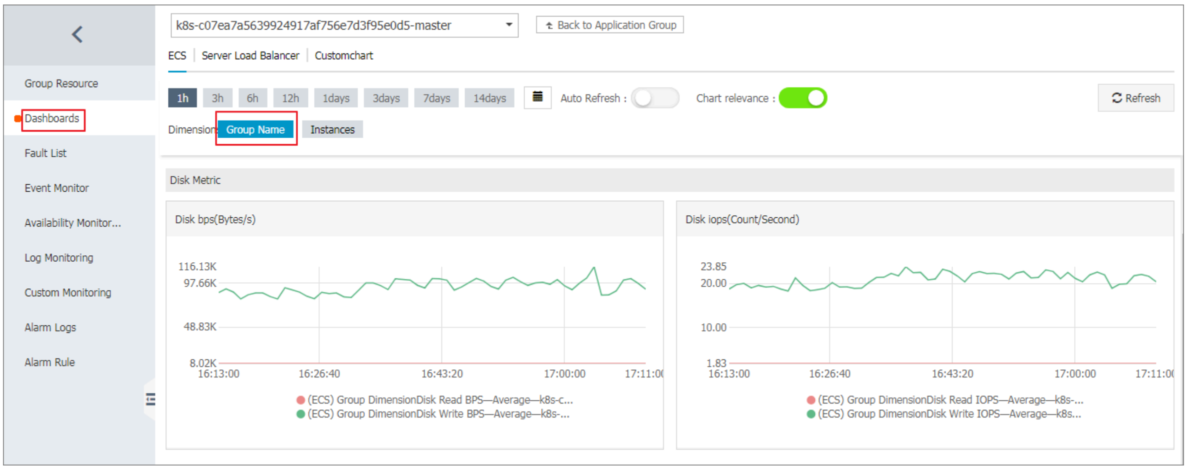


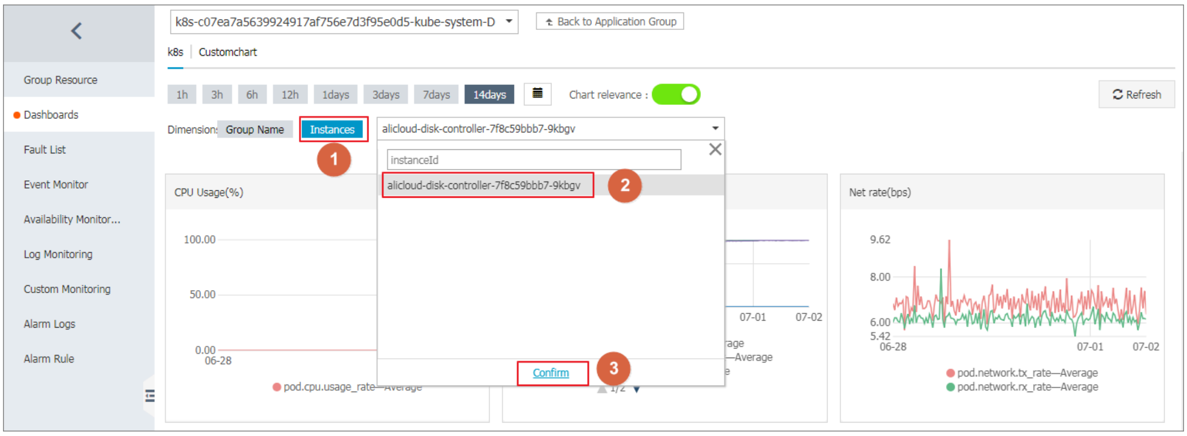


In this case, you jump to the corresponding Application group details page of CloudMonitor.

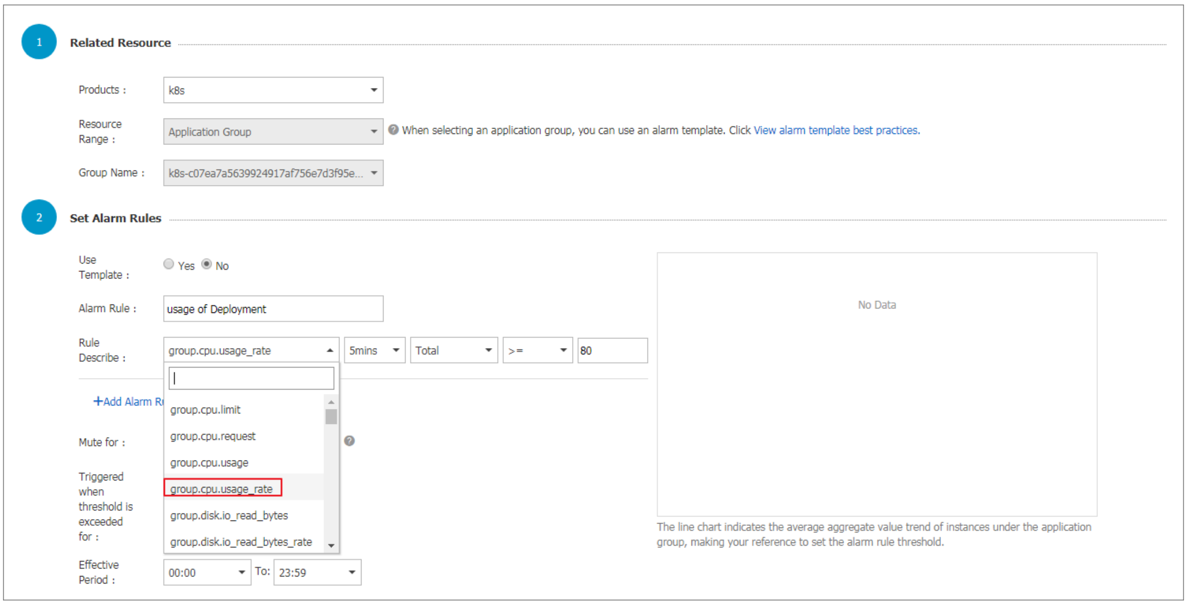


4. Application group supports monitoring in two dimensions: group and instance.





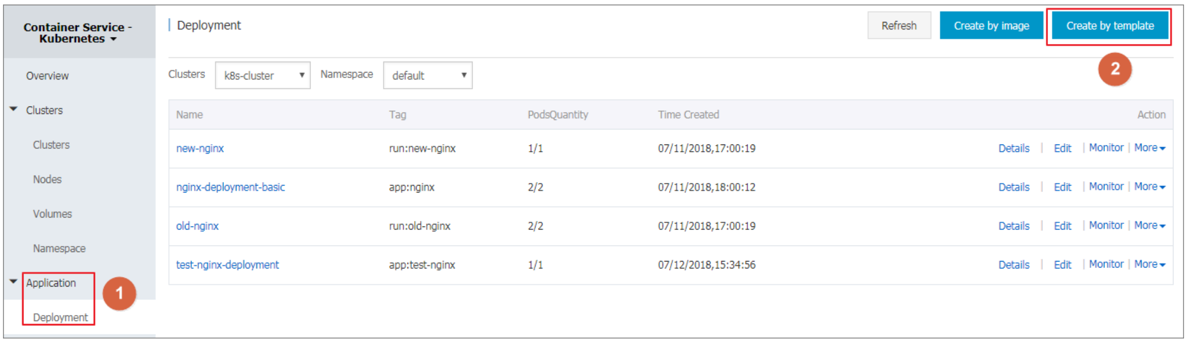
5. For alarm settings, the index of group level starts with `group`, and the instance level index starts with `pod`.



Upgrade cluster version

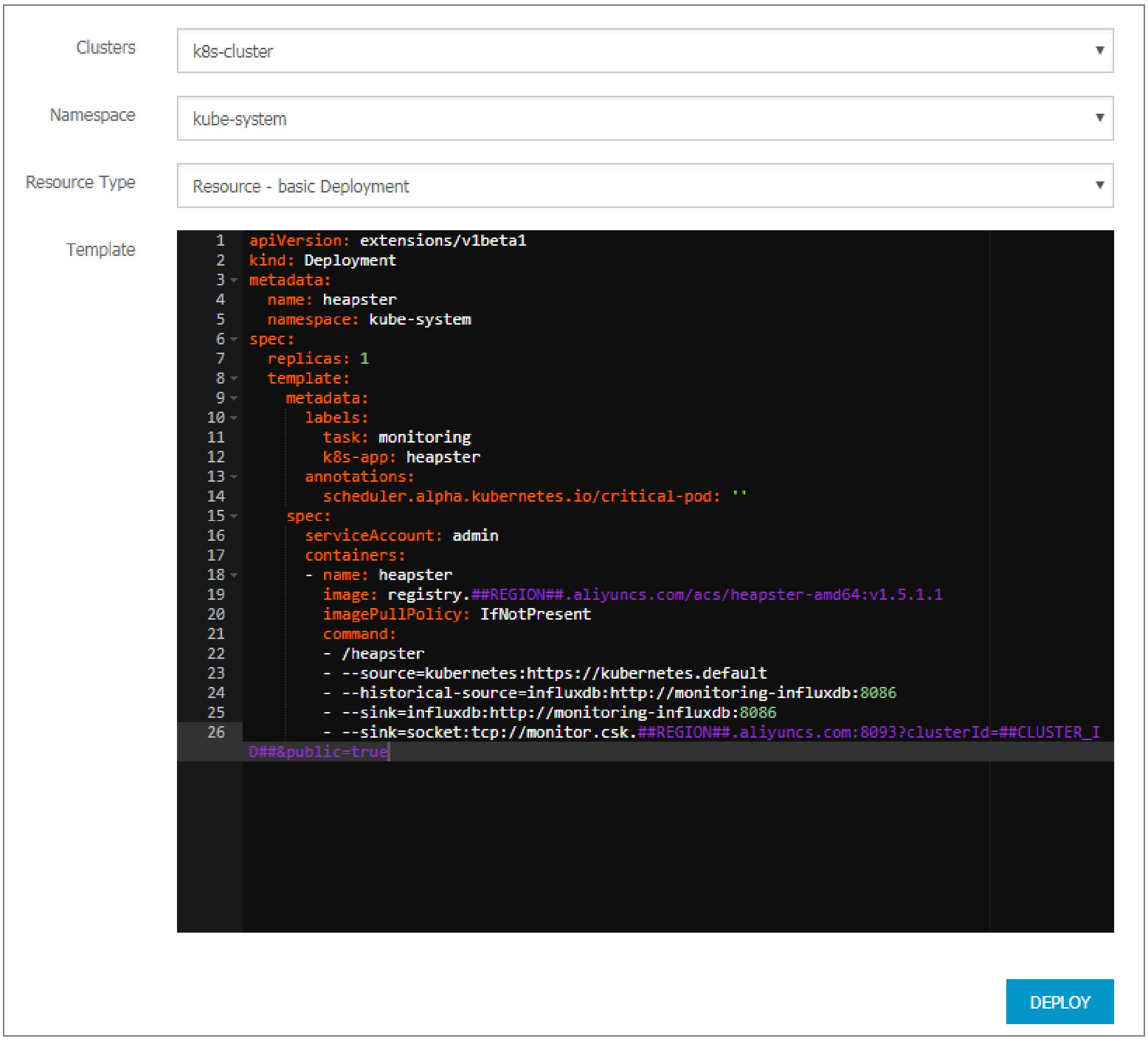
1. Log on to the *Container Service console*.

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.



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.



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.

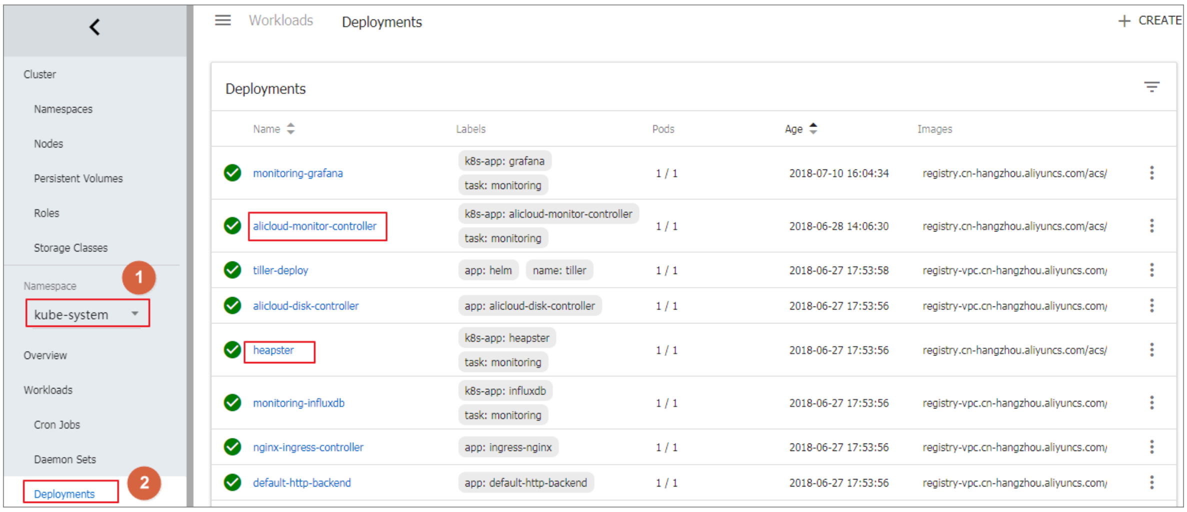
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: heapster
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        task: monitoring
        k8s-app: heapster
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
    spec:
      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
```

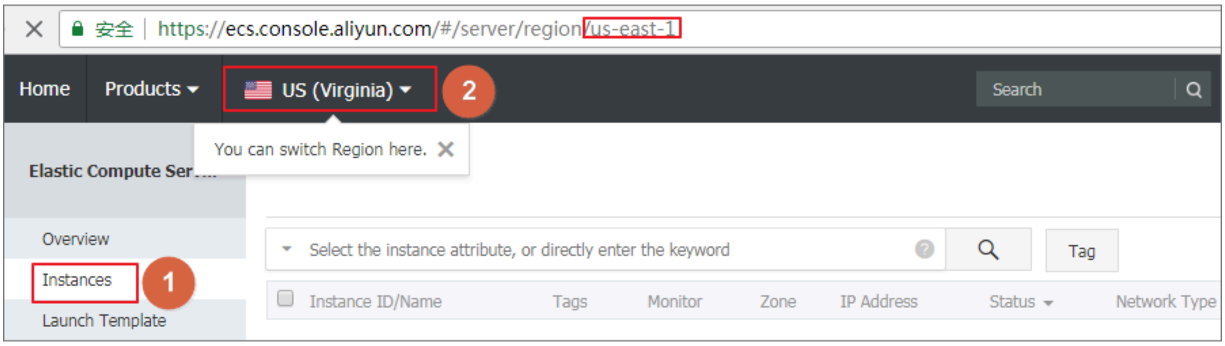
The example layout of alicloud-monitor-controller is as follows. Run the `kubectl create -f xxx.yaml` command to deploy alicloud-monitor-controller.

```
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.



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.



1.13.4 Use Grafana to display monitoring data

Prerequisites

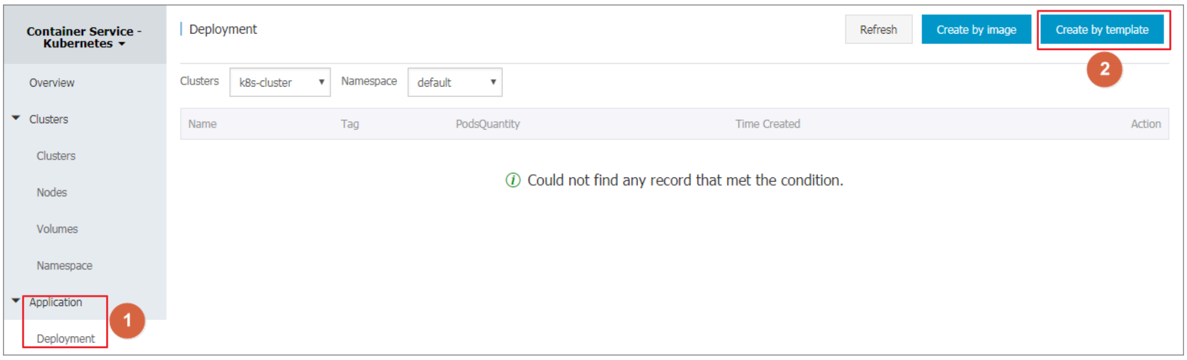
- You have successfully created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- In this example, use the Grafana with built-in monitoring templates and the image address is `registry.cn-hangzhou.aliyuncs.com/acs/grafana:5.0.4`.

Context

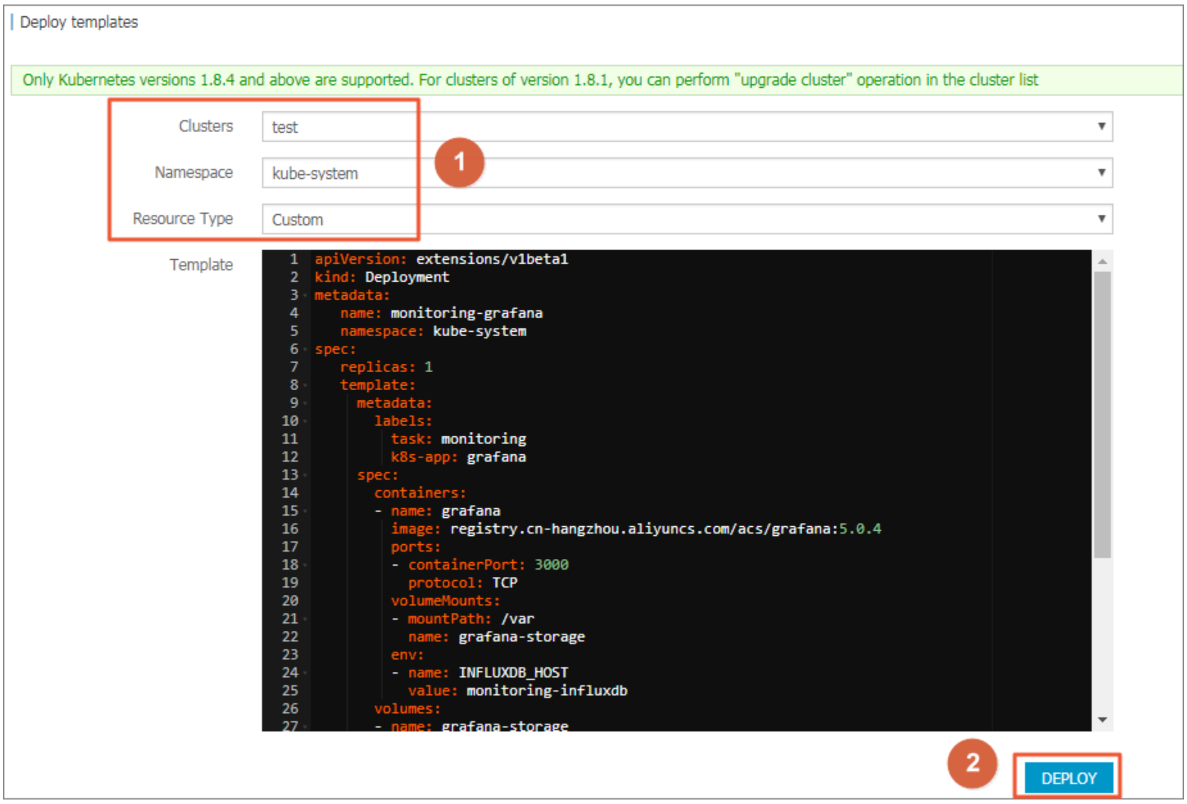
Among Kubernetes monitoring solutions, compared with open-source solutions such as Prometheus, the combination of Heapster + InfluxDB + Grafana is more simple and direct. Heapster not only collects monitoring data in Kubernetes, but also is relied on by the monitoring interface of the console and the 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.



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.

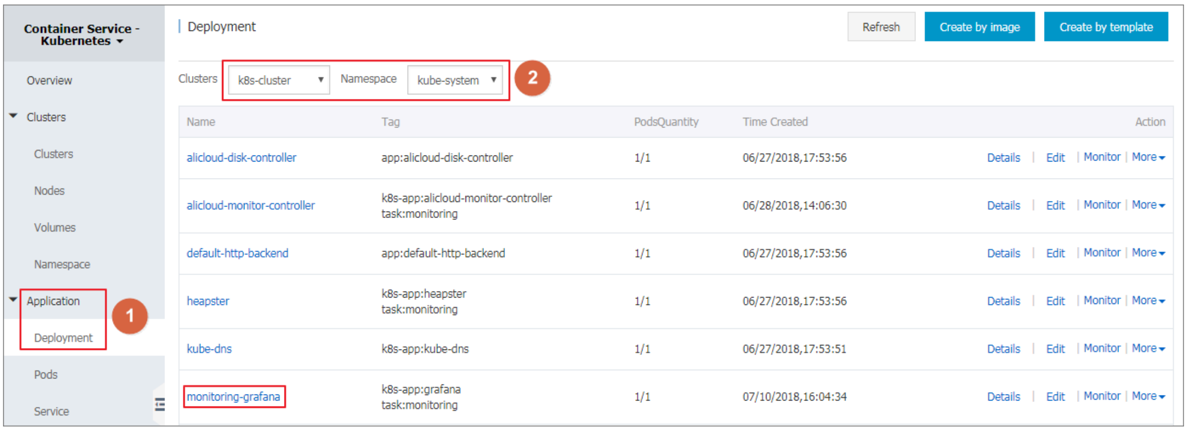


The orchestration template in this example is as follows:

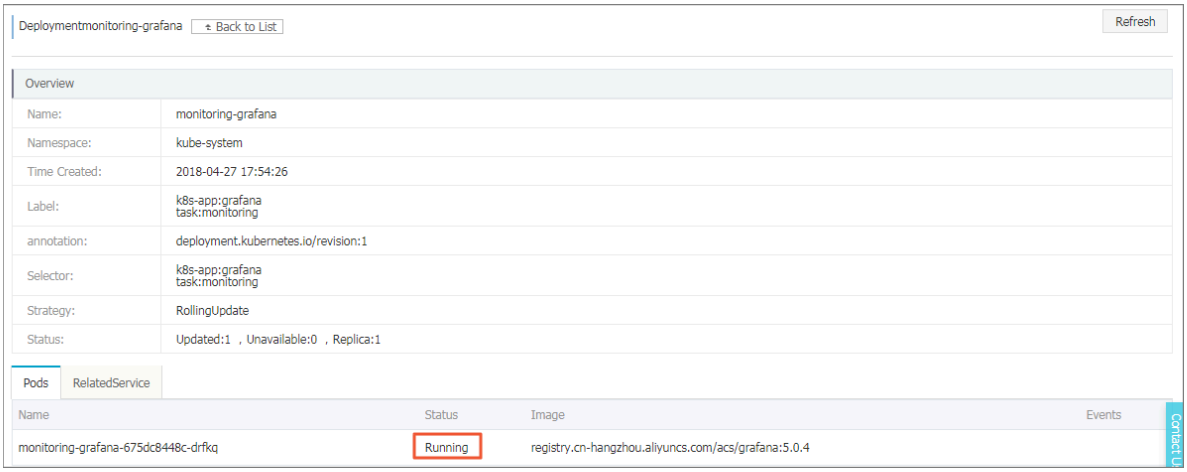
```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
   name: monitoring-grafana
   namespace: kube-system
spec:
   replicas: 1
   template:
     metadata:
       labels:
         task: monitoring
         k8s-app: grafana
     spec:
       containers:
       - name: grafana
         image: 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.

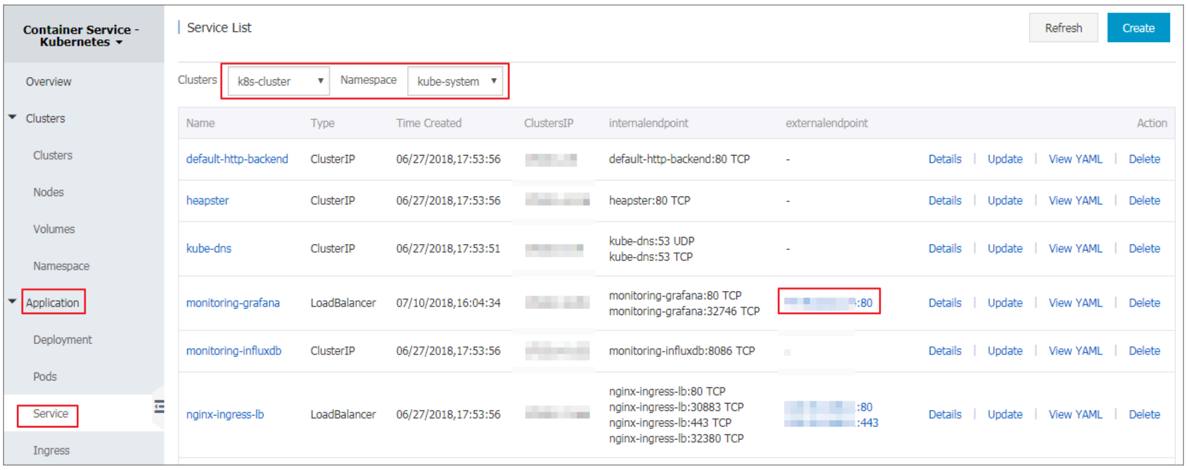


6. Click the name monitoring-grafana to view the deployment status. Wait until the running status changes to Running.



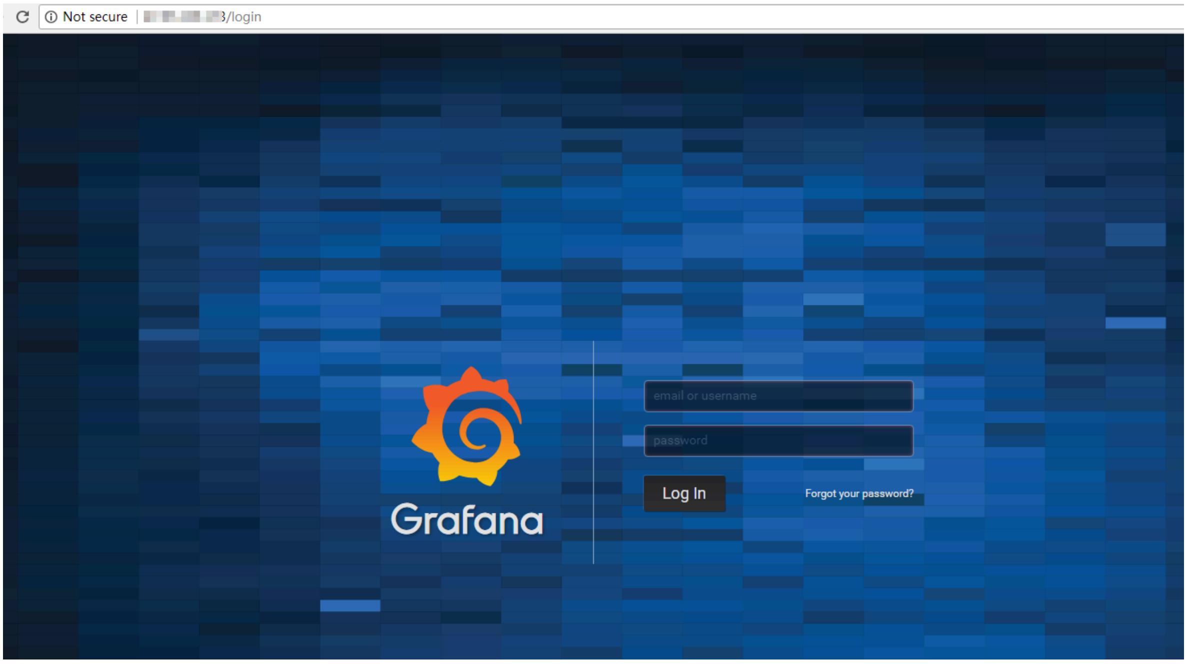
7. Click Application > Service in the left-side navigation pane. Select the cluster from the Clusters drop-down list and kube-system from the Namespace drop-down list to view the external endpoint.

   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.



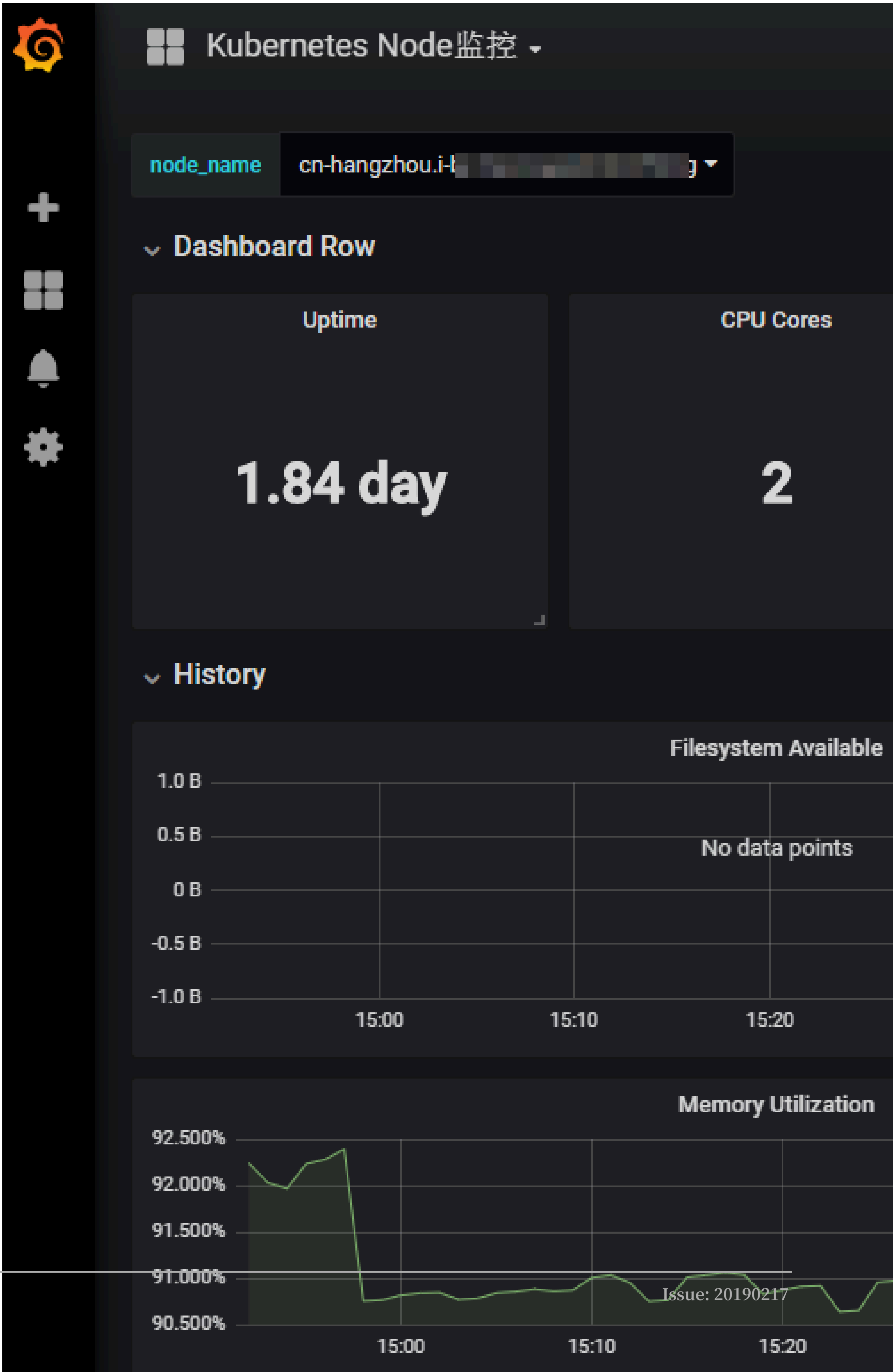
8. Click the external endpoint at the right of the monitoring-grafana service to log on to the Grafana monitoring page.

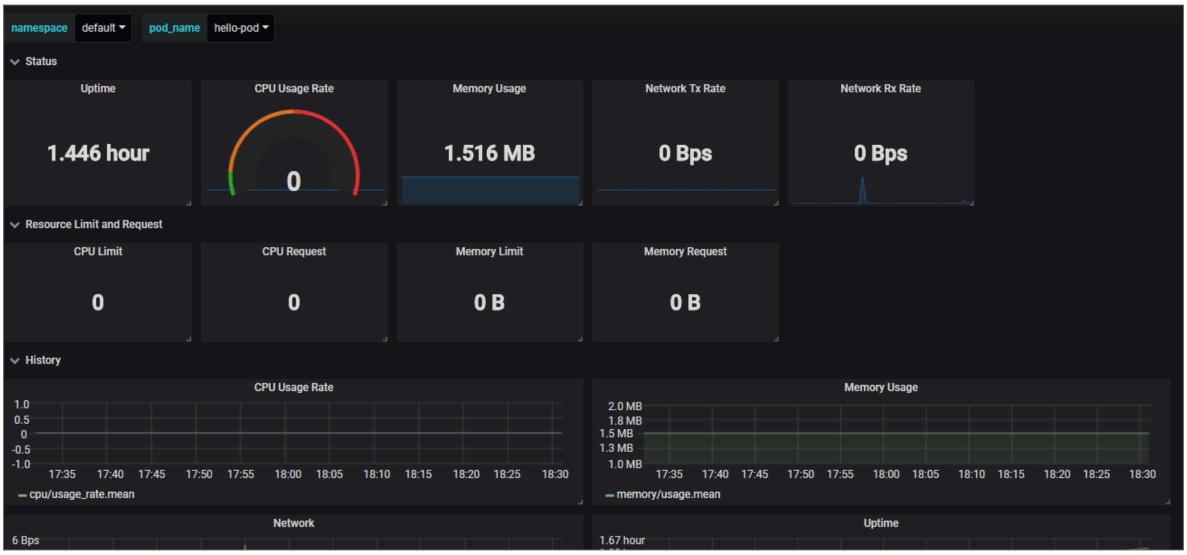
By default, the username and password of Grafana are both admin. We recommend that you change the password after the logon.



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
Filesystem Available
1.0 B
0.5 B
0 B
-0.5 B
-1.0 B
No data points
15:00
15:10
15:20
Memory Utilization
92.500%
92.000%
91.500%
91.000%
90.500%
15:00
15:10
15:20



1.13.5 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).

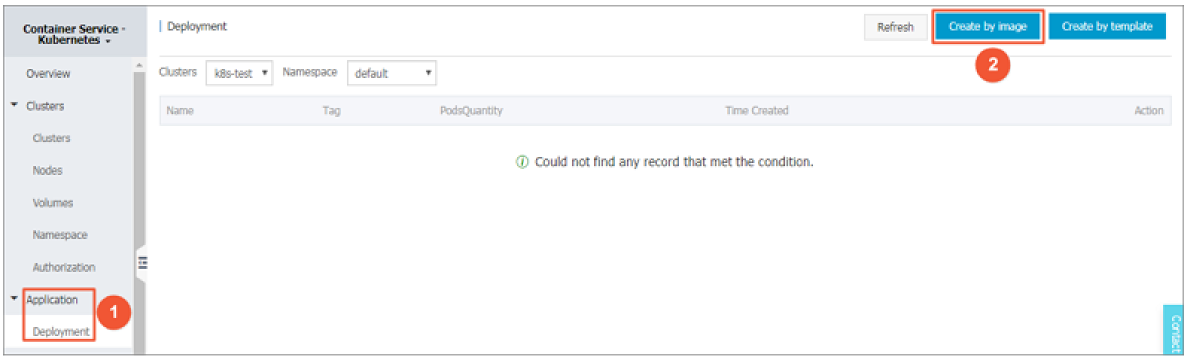
Prerequisites

- 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.

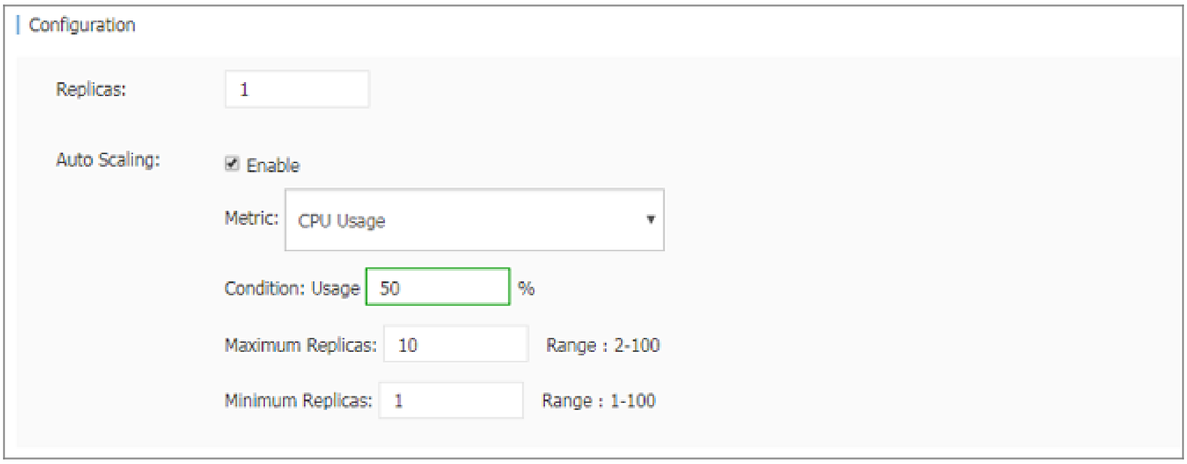
Method 1 Create an HPA application in the Container Service console

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.

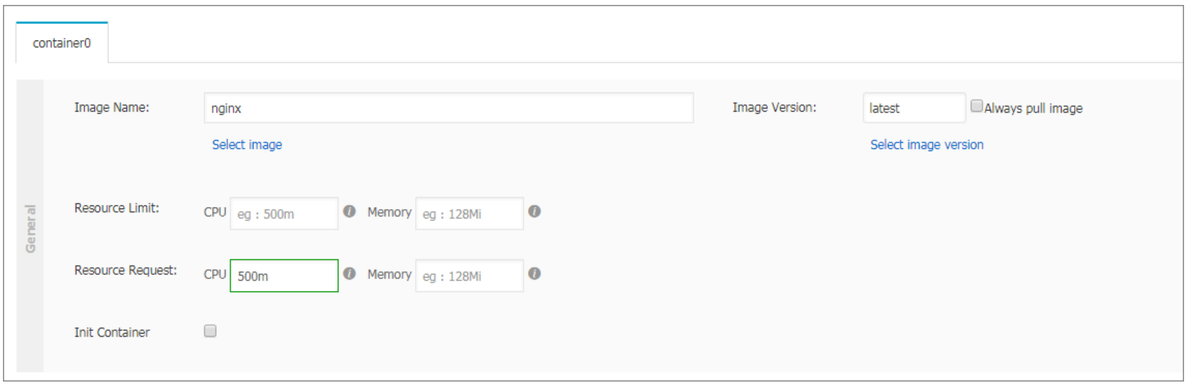


3. Enter the application name, select the cluster and namespace, and click Next.
4. Configure the application settings. Set the number of replicas, select the Enable box for Automatic Scaling, and configure the settings for scaling.
   - Metric: CPU and memory. Configure a resource type as needed.
   - Condition: The percentage value of resource usage. The container begins to expand when the resource usage exceeds this value.
   - Maximum Replicas: The maximum number of replicas that the deployment can expand to.
   - Minimum Replicas: The minimum number of replicas that the deployment can contract to.



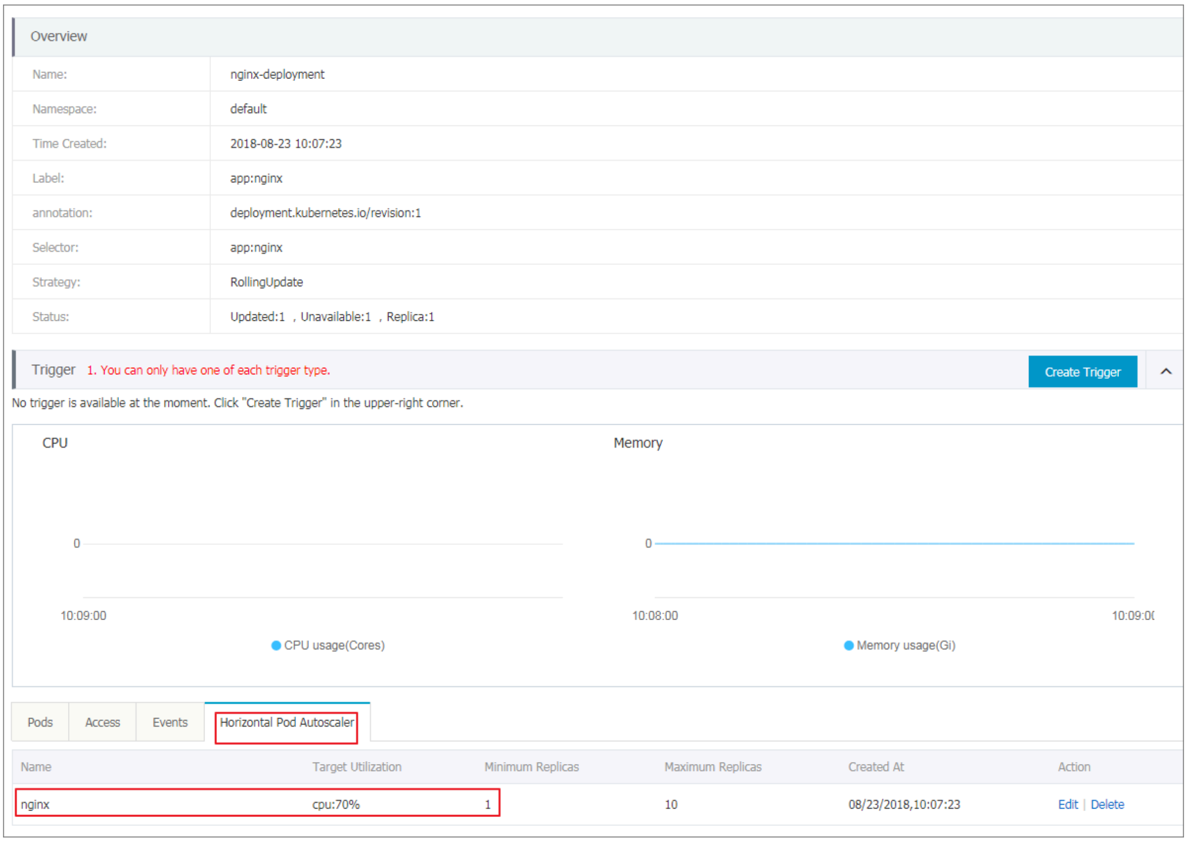
5. Configure the container. Select an image and configure the required resources. Click Next.

Note:
You must configure the required resources for the deployment. Otherwise, container auto scaling cannot be achieved.



6. In the Access Control page, do not configure any settings in this example. Click Create directly.

   Now a deployment that supports HPA has been created. You can view the auto scaling group information in the details of your deployment.



7. In the actual environment, the application scales according to the CPU load. You can also verify auto scaling in the test environment. By performing a CPU pressure test on the pod, you can find that the pod can complete the horizontal expansion in half a minute.

Method 2 Use kubectl commands to configure container auto scaling

You can also manually create an HPA by using an orchestration template and bind it to the deployment object to be scaled. Use the `kubectl` command to complete the container auto scaling configuration.

The following is an example of an Nginx application. Execute the `kubectl create -f xxx.yml` command to create an orchestration template for the deployment as follows:

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    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 horizontally. When the usage of Nginx pod drops below 50%, the container contracts.

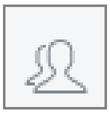
1.13.6 Monitor a Kubernetes cluster and send alarm notifications by using DingTalk

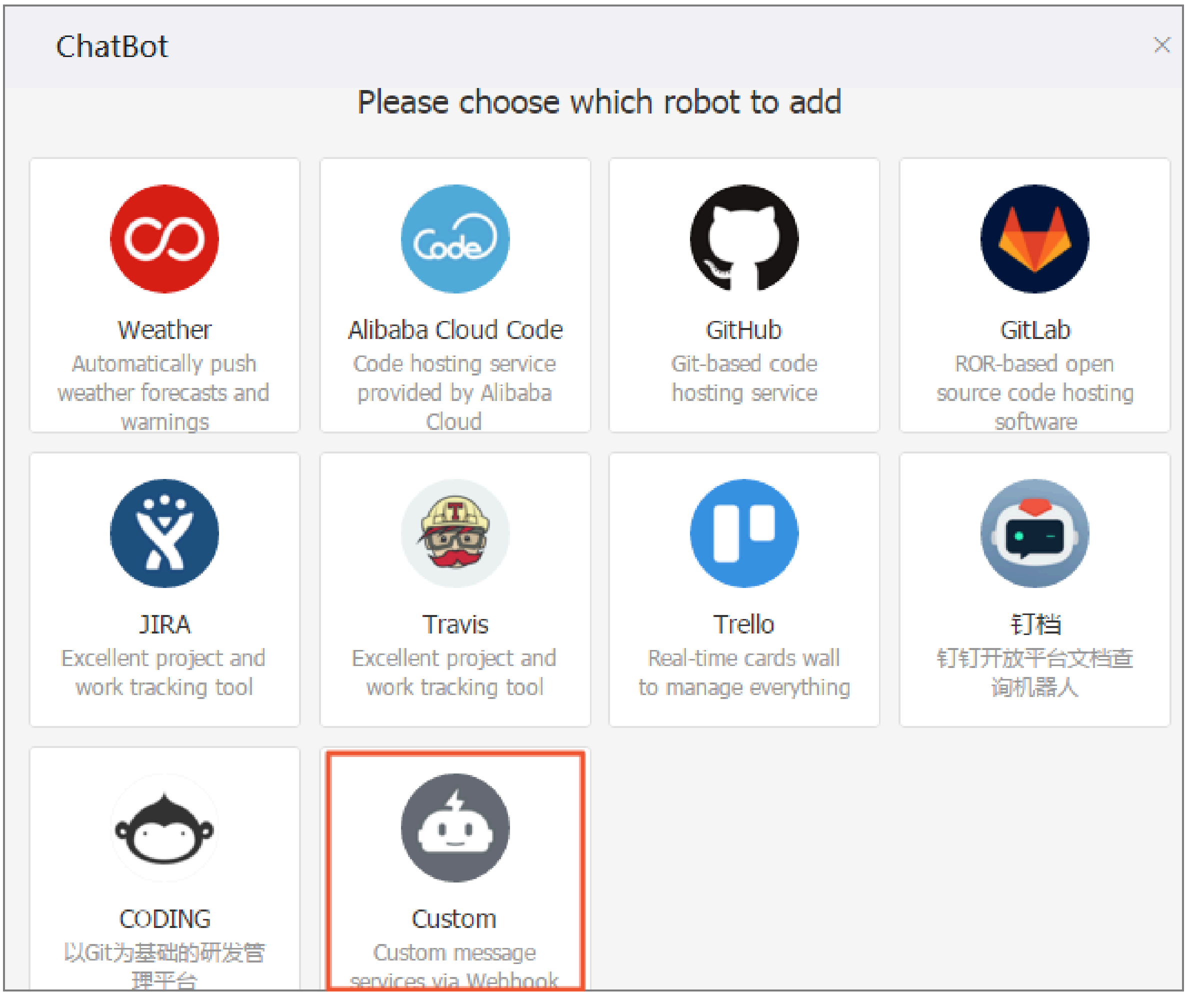
After you deploy a robot in a DingTalk group, the cluster sends a notification of an exception event to the DingTalk group through the robot, implementing real-time monitoring and alarming for cluster exception events.

Context

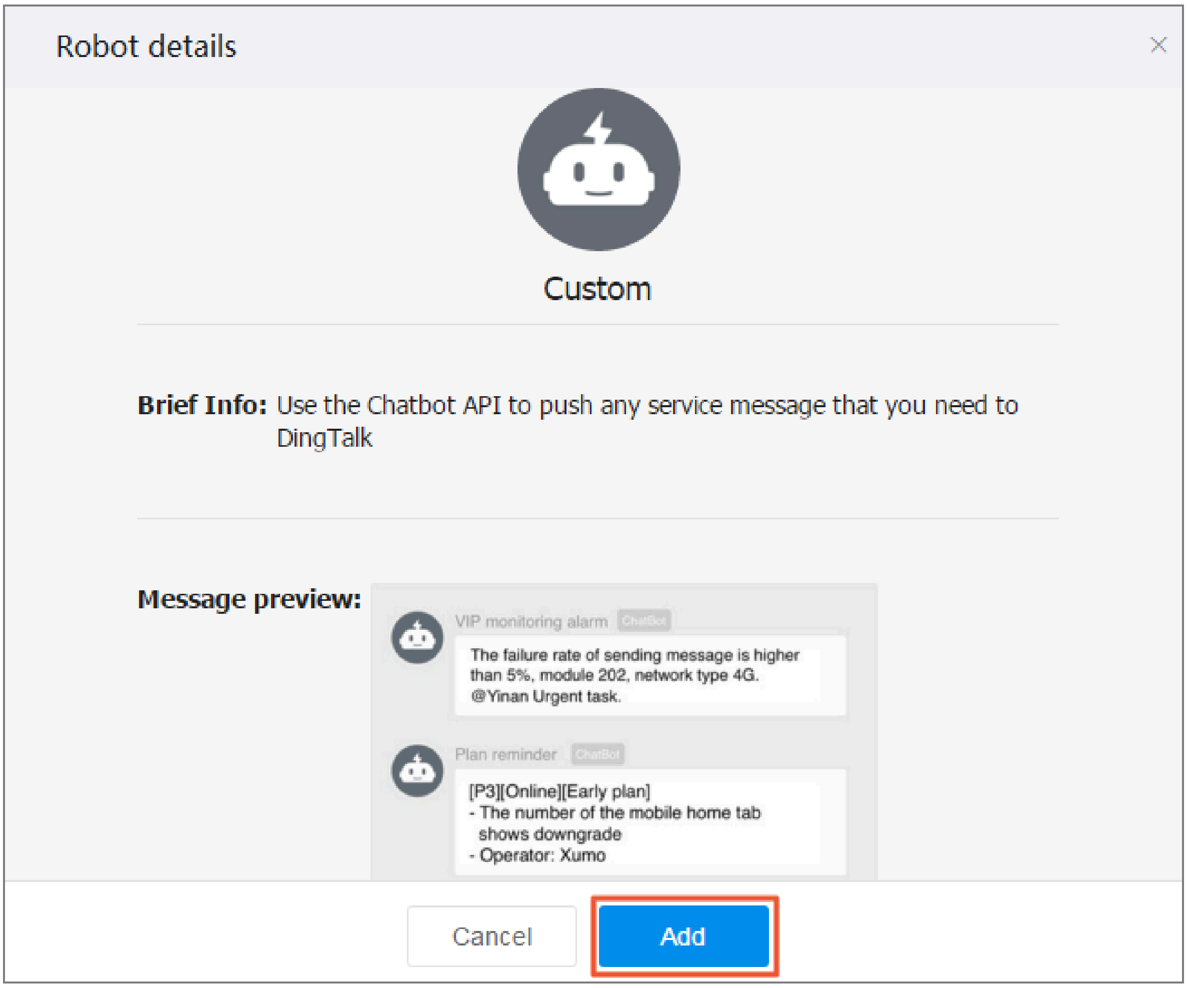
- You have created a DingTalk group .
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

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.



3. On the Robot details page, click Add.

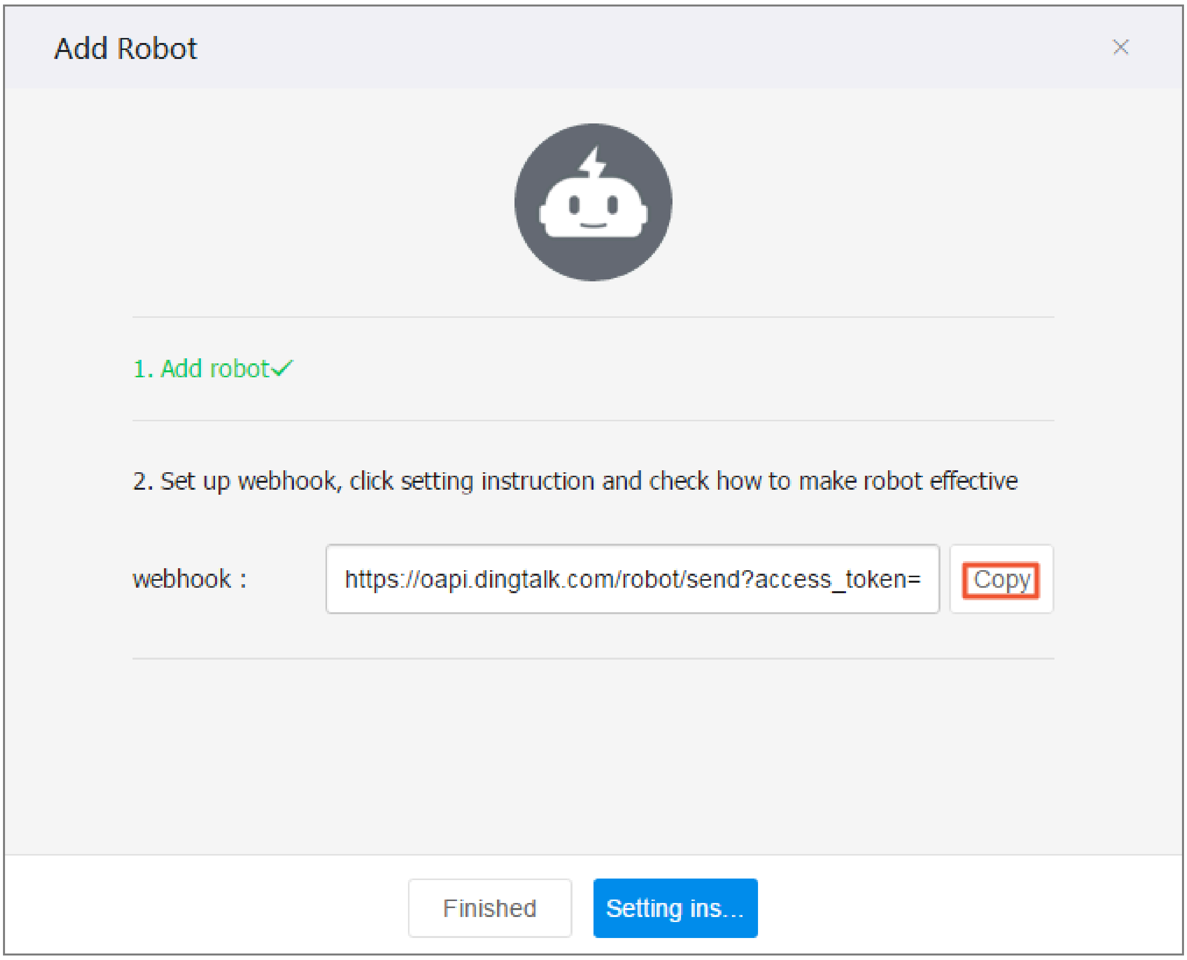


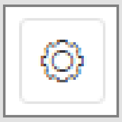
4. Configure the following parameters for a robot and then click Finished:

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: You can configure this parameter after you enable the outgoing function.

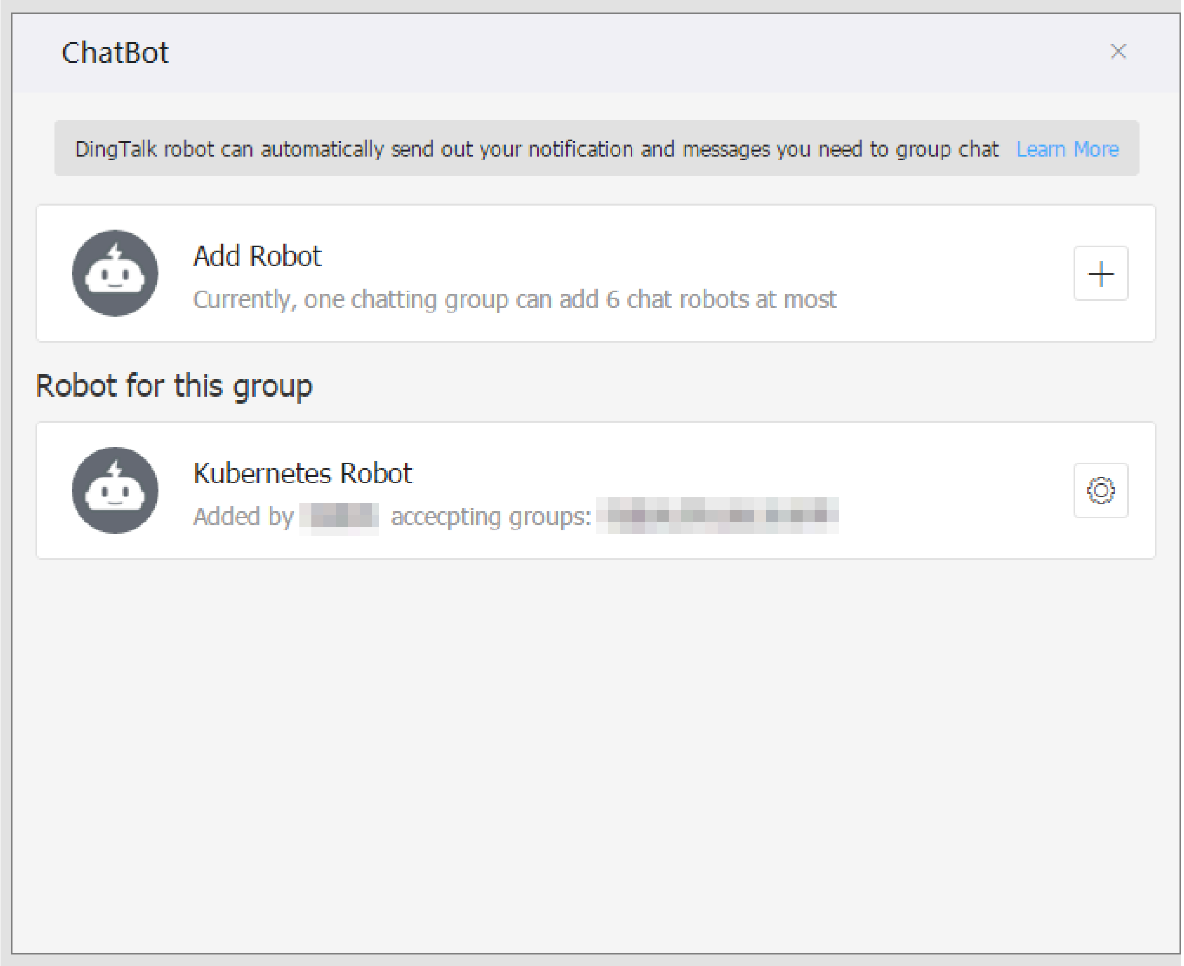
Configuration	Description
Token	The key used to verify that a request is from DingTalk. Note: You can configure this parameter after you enable the outgoing function.

5. Click Copy to copy the webhook address.

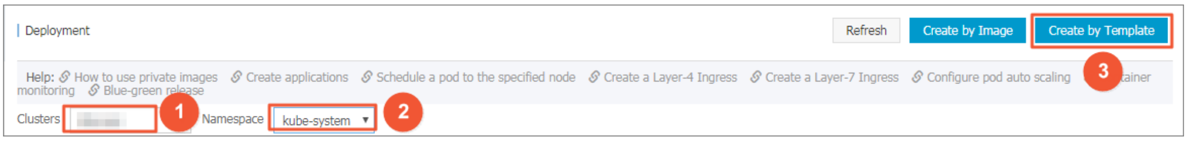


Note:
On the ChatBot page, click the  icon at the right of a robot and then you can perform following operations:

- Modify the profile picture and name of the robot.
- Open or Close notifications.
- Reset the webhook address.
- Remove the robot.



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.



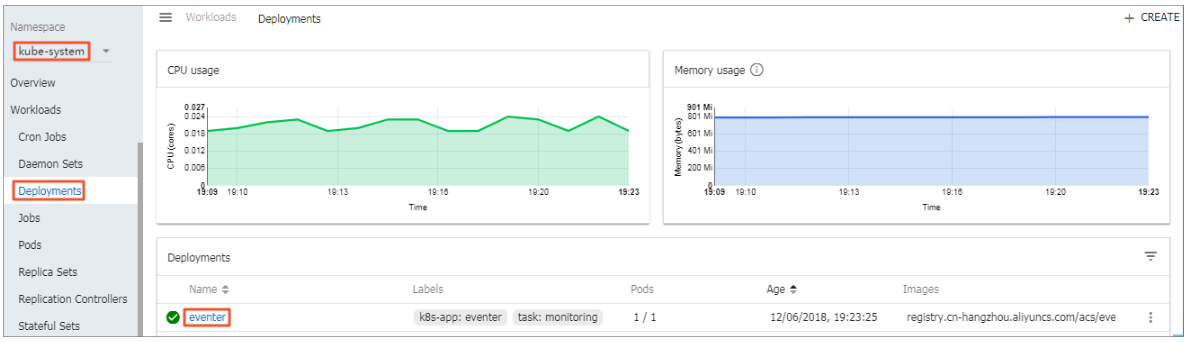
9. Configure a template based on the following parameters, and then click Deploy.

Configuration	Description
Clusters	Select a cluster.
Namespace	Select a namespace to which resource object belongs. The default namespace is default. Select kube-system.

Configuration	Description
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.

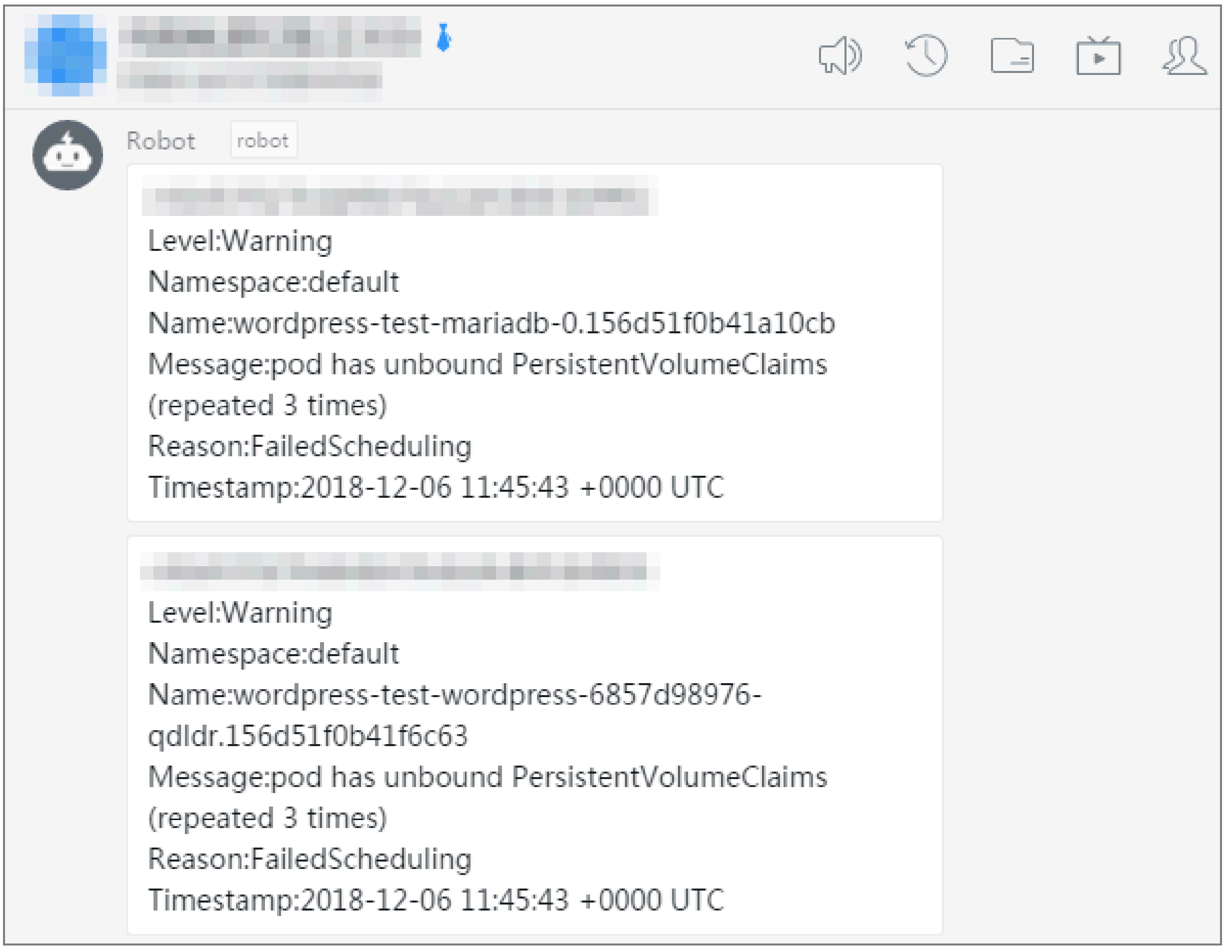
Template	Enter the following custom content:
	``apiVersion: extensions/v1beta1
	kind: Deployment
	metadata:
	  name: eventer
	  namespace: kube-system
	spec:
	  replicas: 1
	  template:
	    metadata:
	      labels:
	        task: monitoring
	        k8s-app: eventer
	      annotations:
	        scheduler.alpha.kubernetes.io/critical-pod: ''
	    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=[Optional parameters are: Normal and Warning.The default is Warning.]
	  #The level field can be set to Normal or Warning. The default is Warning. When the level field is set to Normal, alarm notifications of the Normal and Warning levels can be received in the DingTalk group. When you do not set the level field or set the level field to Warning, only alarm notifications of 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.



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.



1.14 Security management

1.14.1 Security

Authorization

Kubernetes clusters support authorizing RAM users to perform operations on clusters.

For more information, see *Use the Container Service console as a RAM user*.

Full-link TLS certificates

The following communication links in Container Service Kubernetes clusters are verified by TLS certificates to prevent the communication from being eavesdropped or tampered:

- `kubelet` on worker nodes actively communicates with `apiserver` on master nodes
- `apiserver` on master nodes actively communicates with `kubelet` on worker nodes

During initialization, the master node uses SSH tunnels to connect to the SSH service of other nodes (port 22) for initialization.

Native secret & RBAC support

Kubernetes secrets are used to store sensitive information such as passwords, OAuth tokens, and SSH keys. Using plain text to write sensitive information to a pod YAML file or a Docker image may leak the information, while using secrets avoids such security risks effectively.

For more information, see *Secret*.

Role-Based Access Control (RBAC) uses the Kubernetes built-in API group to drive authorization and authentication, which allows you to use APIs to manage pods that correspond to different roles, and the access permissions of roles.

For more information, see *Using RBAC authorization*.

Network policy

In a Kubernetes cluster, pods on different nodes can communicate with each other by default. In some scenarios, to reduce risks, the network intercommunication among different business services is not allowed and you must introduce the network policy. In Kubernetes clusters, you can use the Canal network driver to implement the support for network policy.

Image security scan

Kubernetes clusters can use Container Registry to manage images, which allows you to perform image security scan.

Image security scan identifies the security risks in images quickly and reduces the possibility of applications running on your Kubernetes cluster being attacked.

For more information, see *Image security scan*.

Security group and Internet access

By default, each newly created Kubernetes cluster is assigned a new security group with the minimal security risk. This security group only allows ICMP for the Internet inbound.

By default, you cannot use Internet SSH to access your clusters. To use Internet SSH to connect to the cluster nodes, see *Access Kubernetes clusters by using SSH*.

The cluster nodes access the Internet by using the NAT Gateway, which further reduces the security risks.

1.14.2 Kube-apiserver audit logs

In a Kubernetes cluster, apiserver audit logs record daily operations of different users for you to trace and play an important part in the Operation & Maintenance (O&M) security of the cluster. This topic introduces the configurations of apiserver audit logs of an Alibaba Cloud Kubernetes cluster, and describes how to collect and search logs by using Log Service.

Configurations of apiserver audit logs

Currently, the apiserver audit function is enabled by default when you create a Kubernetes cluster. Relevant parameters and description are as follows:

> **Note:**
> Log on to the Master node, and the directory of the apiserver configuration files is `/etc/kubernetes/manifests/kube-apiserver.yaml`.

Configuration	Description
audit-log-maxbackup	The maximum fragment of audit logs stores 10 log files.

Configuration	Description
audit-log-maxsize	The maximum size of a single audit log is 100 MB.
audit-log-path	The audit log output path is */var/log/kubernetes/kubernetes.audit*.
audit-log-maxage	The longest storage period of audit logs is seven days.
audit-policy-file	Configuration policy file of audit logs. The directory is */etc/kubernetes/audit-policy.yml*.

Log on to the Master node machine. The directory of the audit log configuration policy file is */etc/kubernetes/audit-policy.yml*. The content of the file is as follows:

```
apiVersion: audit.k8s.io/v1beta1 # This is required.
kind: Policy
# We recommend that you do not generate audit events for all requests
in RequestReceived stage.
omitStages:
  - "RequestReceived"
rules:
  # The following requests are manually identified as high-volume and
low-risk.
  # Therefore, we recommend that you drop them.
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
      - group: "" # core
        resources: ["endpoints", "services"]
  - level: None
    users: ["system:unsecured"]
    namespaces: ["kube-system"]
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["configmaps"]
  - level: None
    users: ["kubelet"] # legacy kubelet identity
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["nodes"]
  - level: None
    userGroups: ["system:nodes"]
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["nodes"]
  - level: None
    users:
      - system:kube-controller-manager
```

```
      - system:kube-scheduler
      - system:serviceaccount:kube-system:endpoint-controller
    verbs: ["get", "update"]
    namespaces: ["kube-system"]
    resources:
      - group: "" # core
        resources: ["endpoints"]
  - level: None
    users: ["system:apiserver"]
    verbs: ["get"]
    resources:
      - group: "" # core
        resources: ["namespaces"]
  # We recommend that you do not log these read-only URLs.
  - level: None
    nonResourceURLs:
      - /healthz*
      - /version
      - /swagger*
  # We recommend that you do not log events requests.
  - level: None
    resources:
      - group: "" # core
        resources: ["events"]
  # Secrets, ConfigMaps, and TokenReviews can contain sensitive and
binary data.
  # Therefore, they are logged only at the Metadata level.
  - level: Metadata
    resources:
      - group: "" # core
        resources: ["secrets", "configmaps"]
      - group: authentication.k8s.io
        resources: ["tokenreviews"]
  # Get repsonses can be large; skip them.
  - level: Request
    verbs: ["get", "list", "watch"]
    resources:
      - group: "" # core
      - group: "admissionregistration.k8s.io"
      - group: "apps"
      - group: "authentication.k8s.io"
      - group: "authorization.k8s.io"
      - group: "autoscaling"
      - group: "batch"
      - group: "certificates.k8s.io"
      - group: "extensions"
      - group: "networking.k8s.io"
      - group: "policy"
      - group: "rbac.authorization.k8s.io"
      - group: "settings.k8s.io"
      - group: "storage.k8s.io"
  # Default level for known APIs.
  - level: RequestResponse
    resources:
      - group: "" # core
      - group: "admissionregistration.k8s.io"
      - group: "apps"
      - group: "authentication.k8s.io"
      - group: "authorization.k8s.io"
      - group: "autoscaling"
      - group: "batch"
      - group: "certificates.k8s.io"
      - group: "extensions"
      - group: "networking.k8s.io"
```

```
        - group: "policy"
        - group: "rbac.authorization.k8s.io"
        - group: "settings.k8s.io"
        - group: "storage.k8s.io"
    # Default level for all other requests.
    - level: Metadata
```

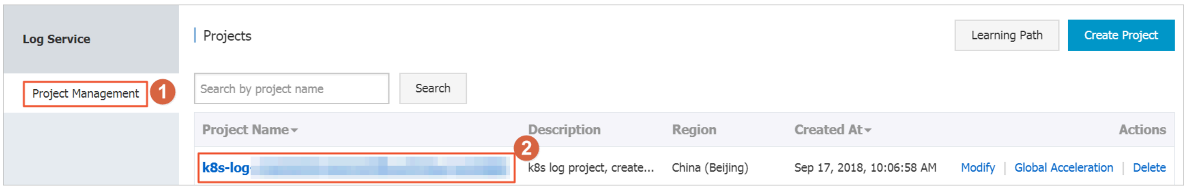
Note:

- Logs are not recorded immediately after requests are received. Log recording starts only after the response body header is sent.
- The following requests or operations are not audited: redundant kube-proxy watch requests, GET requests from kubelet and system:nodes for nodes, operations performed on endpoints by kube components in the kube-system, and GET requests from the apiserver for namespaces.
- Read-only urls such as `/healthz*`, `/version*`, and `/swagger*` are not audited.
- Logs of interfaces of secrets, configmaps, and tokenreviews are set to the metadata level because they might contain sensitive information or binary files . For logs of this level, only the user, timestamp, request resources, and request actions of the request event are audited. The request body and the response body are not audited.
- For sensitive interfaces such as authentication, rbac, certificates, autoscaling, and storage, the corresponding request bodies and response bodies are audited according to the read and write requests.

Collect and search logs

Before you use Kube-apiserver audit logs, make sure that Log Service is enabled when you create a cluster and the corresponding log Project and Logstore are created.

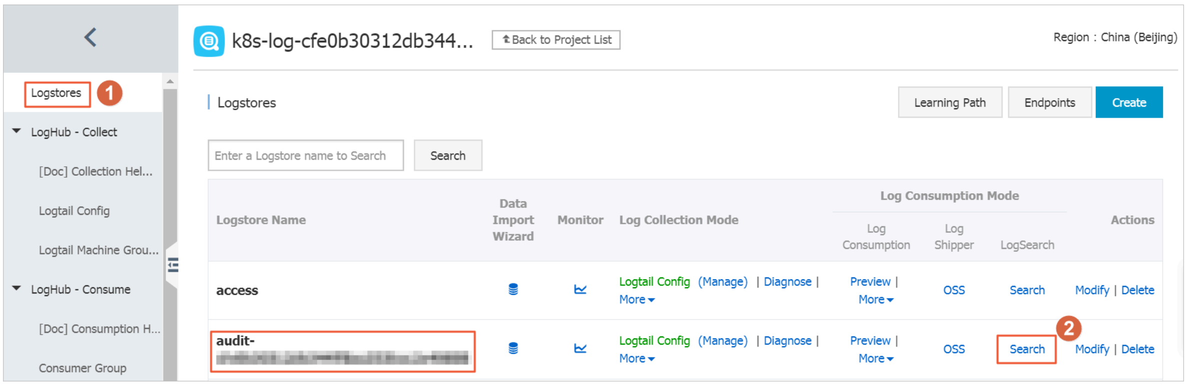
1. Log on to the *Log Service console*.
2. In the left-side navigation pane, click Project Management, select the Project configured when you create the cluster, and then click the Project name.



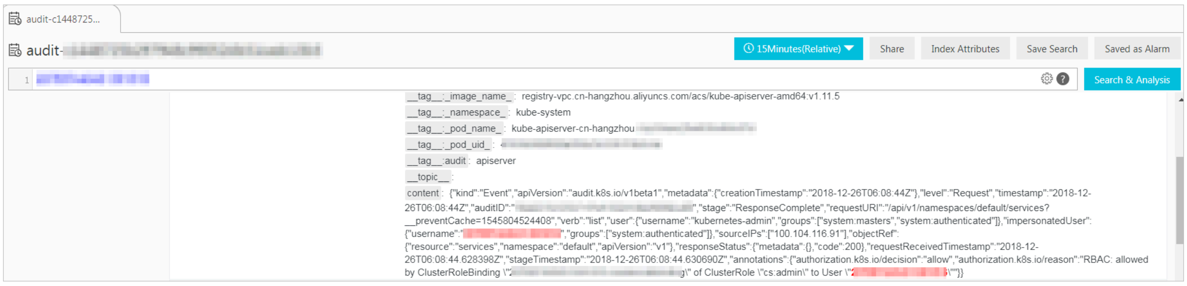
3. On the Logstores page, find the Logstore named audit-${clusterid} and click Search at the right side of the Logstore. The audit logs of the cluster are stored in this Logstore.

Note:

In the process to create the cluster, a Logstore named audit-${clusterid} is automatically added to your specified Project.



4. To trace the operations of a RAM user, enter the RAM user ID to search relevant logs, as shown in the following figure.



5. To trace the operations on a specific resource object in a specified period of time, enter the resource name to search relevant logs, as shown in the following figure.

Use a thirty-party log solution

Log on to the Master node of the cluster, and you can find the source file of the audit logs in the path of `/var/log/kubernetes/kubernetes.audit`. The source file is in standard json format. When deploying a cluster, you can use other log solutions to collect and search audit logs, instead of using Alibaba Cloud Log Service.

1.14.3 Implement secure access through HTTPS in Kubernetes

A Container Service Kubernetes cluster supports multiple application access methods. The most common methods include `SLB:Port` access, `NodeIP:NodePort` access, and domain name access. By default, a Kubernetes cluster does not support HTTPS access. To access applications through HTTPS, you can use the secure HTTPS access method provided by Container Service and Alibaba Cloud Server Load Balancer (SLB) service. This document explains how to configure a certificate in Container Service Kubernetes by using HTTPS access configuration as an example.

Depending on different access methods, your certificate can be configured with the following two methods:

- Configure the certificate on the frontend SLB.
- Configure the certificate on Ingress.

Prerequisites

- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have connected to the Master node through SSH. For more information, see *Access Kubernetes clusters by using SSH*.
- After connecting to the Master node, you have created the server certificates for the cluster, including the public key certificate and the private key certificate by running the following commands :

```
$  openssl genrsa -out tls.key 2048

Generating RSA private key, 2048 bit long modulus
................................................................ +++
..............................................................................
+++
e is 65537 (0x10001)

$  openssl req -sha256 -new -x509 -days 365 -key tls.key -out tls.
crt

You are about to be asked to enter 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:

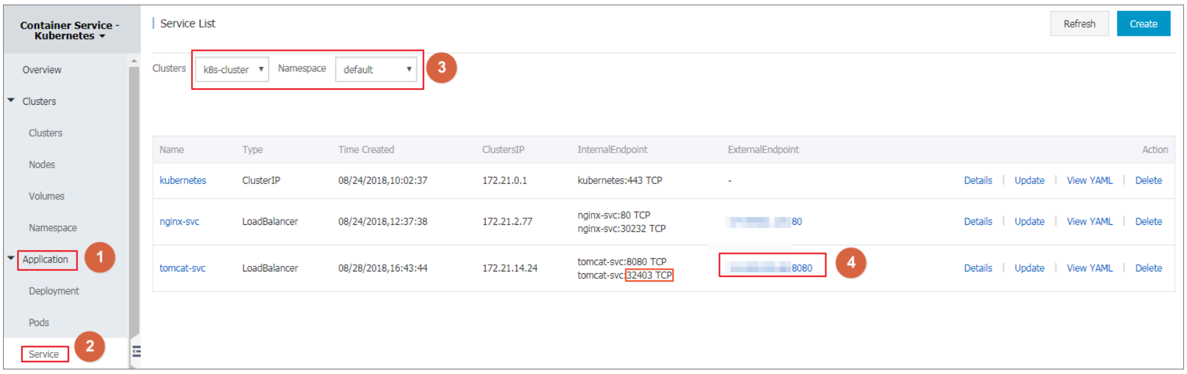
- Advantages: The certificate is configured on SLB and it is the external access portal of applications. The access to applications in the cluster still uses the HTTP access method.
- Disadvantages: You need to maintain many associations between domain names and their corresponding IP addresses.
- Scenarios: This method is applicable to applications that use LoadBalancer service rather than Ingress to expose access methods.

Preparations

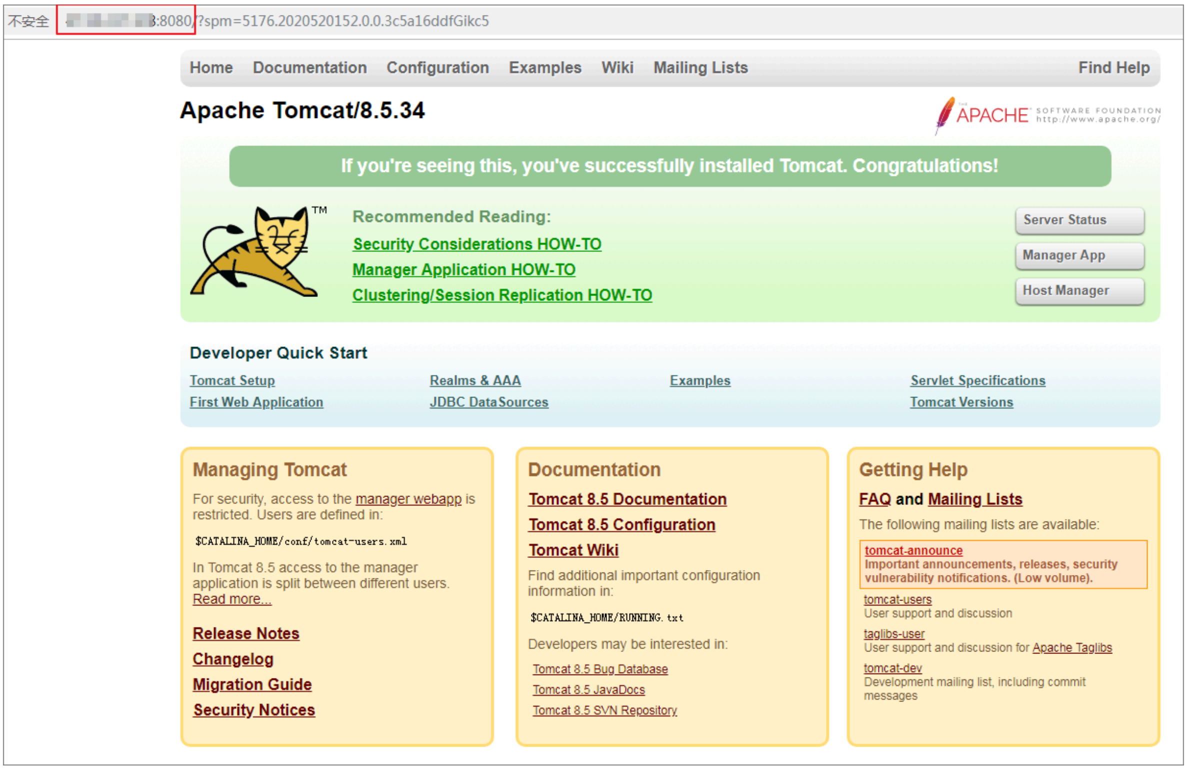
You have created a Tomcat application in the Kubernetes cluster. The application provides external access by using the LoadBalancer service. For more information, see *Create a service*.

Example

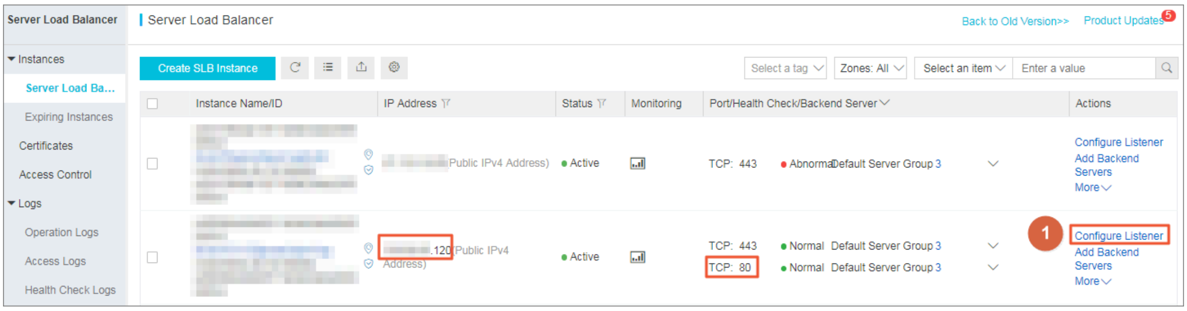
1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click Application > Service, and select the cluster and the namespace to view the pre-created Tomcat application. As shown in the following figure, the created Tomcat application is named tomcat and the service name is tomcat-svc. The service type of the application is LoadBalancer, and the service port exposed by the application is 8080.



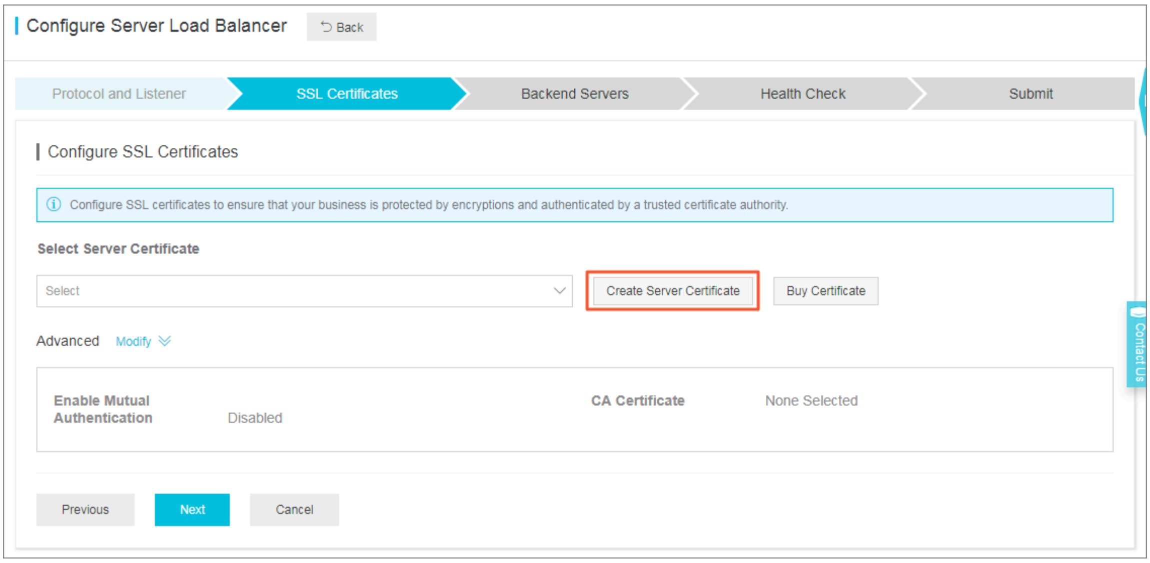
3. By clicking the external endpoint, you can access the Tomcat application through `IP:Port`.



4. Log on to the *SLB console*.
5. By default, the Server Load Balancer page is displayed. In the IP address column, find the server load balancer that corresponds to the external endpoint of the tomcat-svc service, and click Configure Listener in the actions column.

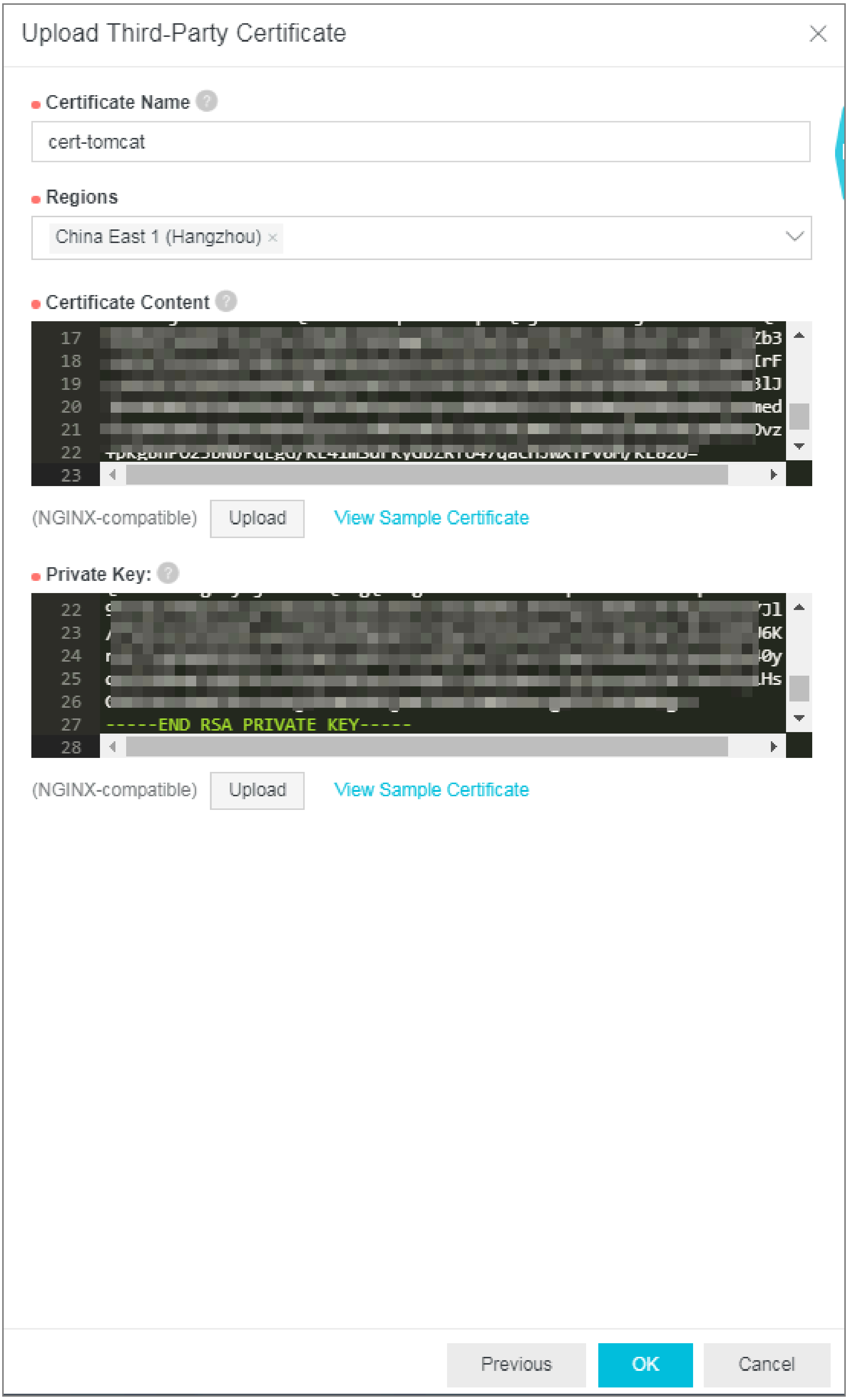


6. Configure the server load balancer. Select a listener protocol first. Select HTTPS, set the listening port to 443, and then click Next.
7. Configure the SSL certificate.
   a. Click Create Server Certificate.



b. On the displayed page, select a certificate source. In this example, select Upload Third-Party Certificate, and then click Next.

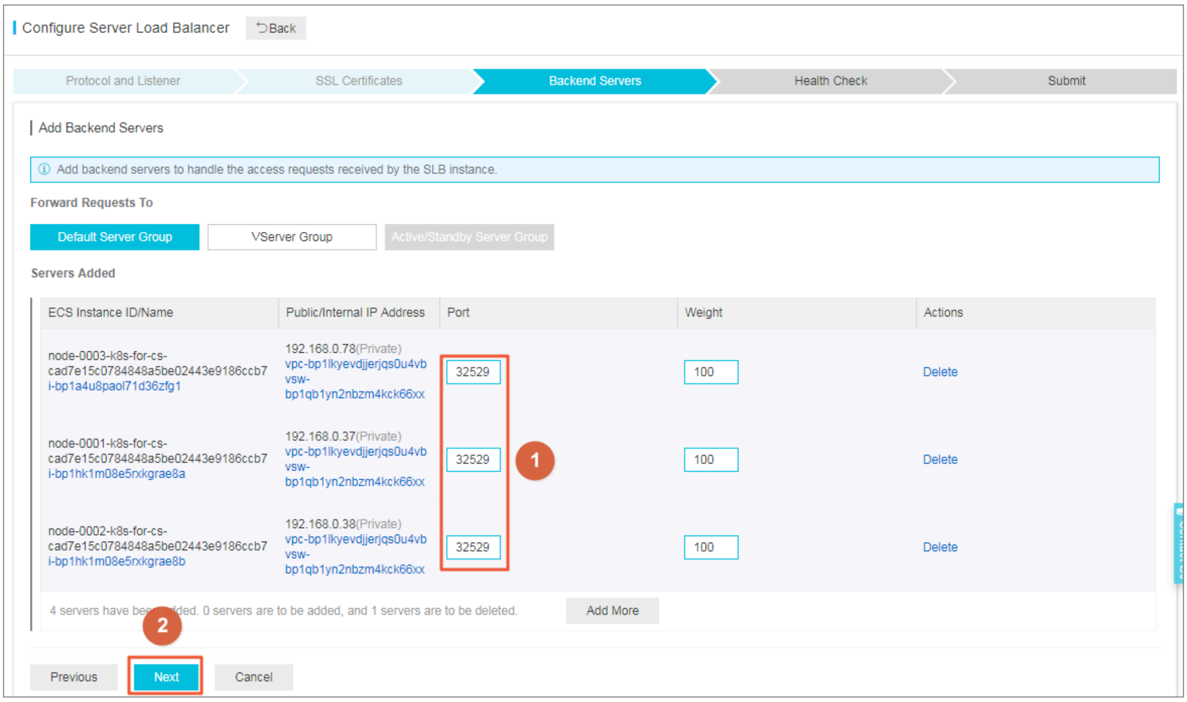
c. On the uploading third-party certificate page, set the certificate name and select the region in which the certificate is deployed. In the Certificate Content and the Private Key columns, enter the server public key certificate and private key created in *Prerequisites*, and then click OK.


Upload Third-Party Certificate
Certificate Name
cert-tomcat
Regions
China East 1 (Hangzhou)
Certificate Content
(NGINX-compatible)
Upload
View Sample Certificate
Private Key:
-----END RSA PRIVATE KEY-----
(NGINX-compatible)
Upload
View Sample Certificate
Previous
OK
Cancel

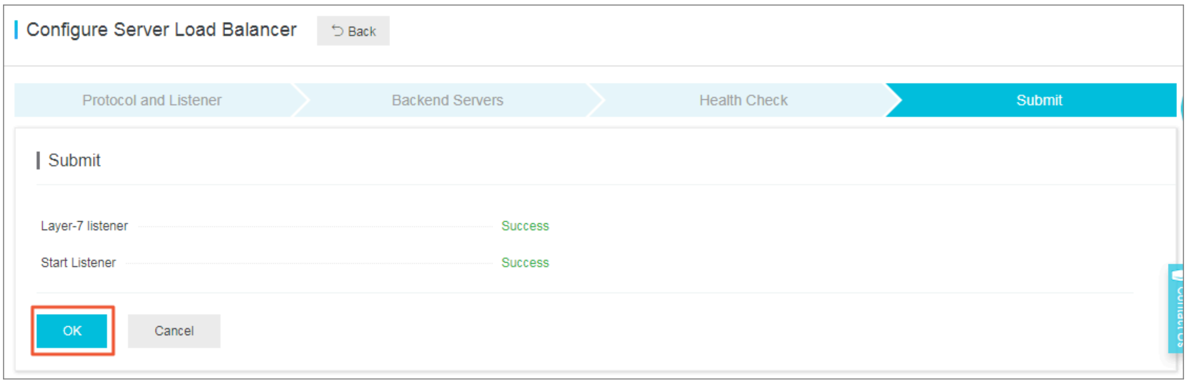
d. From the Select Server Certificate drop-down list, select the created server certificate.
e. Click Next.

8. Configure Backend Servers. By default, servers are added. You need to configure a port for each backend server to listen to the tomcat-svc service, and then click Next.

> **Note:**
> You need to find the NodePort number of this service in the Container Service Web interface, and configure the number as the port number of each backend server.



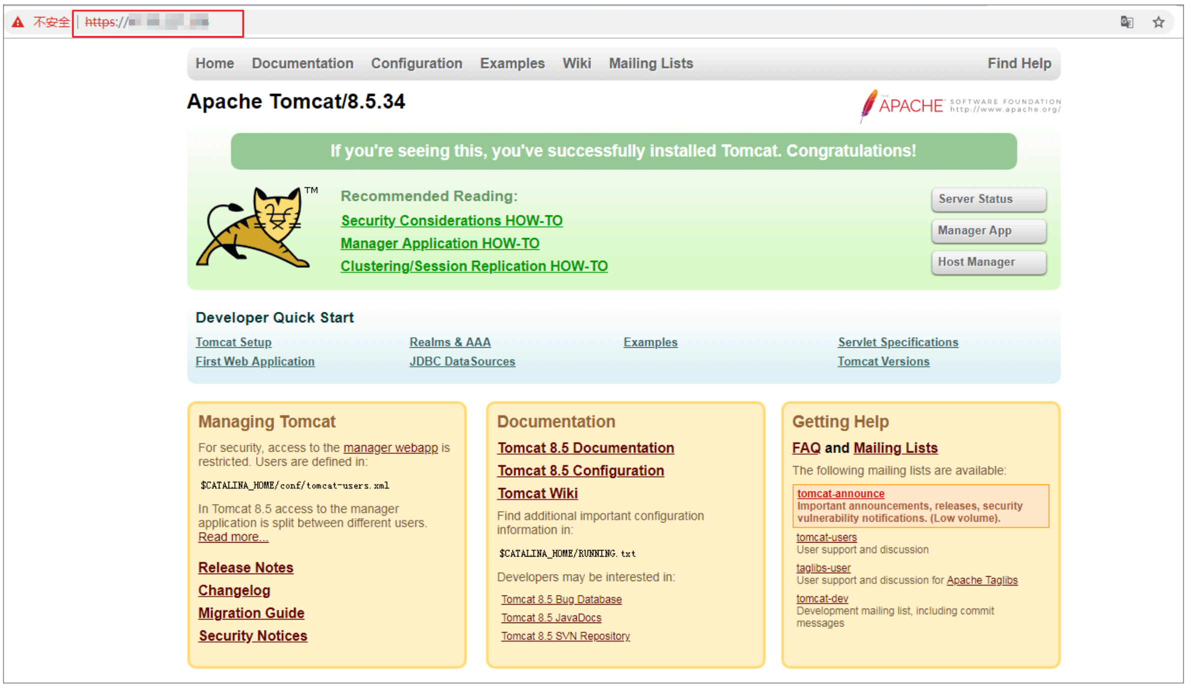
9. Configure Health Check, and then click Next. In this example, use the default settings.
10. Confirm the Submit tab. When you make sure that all configurations are correct, click Submit.
11. After completing the configuration, click OK.



12.Return to the Server Load Balancer page to view the instance. The listening rule of `HTTPS:443`is generated.

13.Access the Tomcat application through HTTPS. In the address bar of the browser, enter `https://slb_ip` to access the application.

> **Note:**
> If the domain name authentication is included in the certificate, you can access the application by using the domain name. You can also access the application through `slb_ip:8080` because `tcp:8080` is not deleted.



Method 2: Configure the certificate on Ingress

This method has the following advantages and disadvantages:

- Advantages: You do not need to modify the SLB configuration. All applications can manage their own certificates through Ingress without interfering with each other.

- Disadvantages: Each application can be accessed by using a separate certificate or the cluster has applications that can be accessed by only using a certificate.

Preparations

You have created a Tomcat application in the Kubernetes cluster. The service of the application provides access through ClusterIP. In this example, use Ingress to provide the HTTPS access service.

Example

1. Log on to the Master node of the Kubernetes cluster and create a secret according to the prepared certificate.

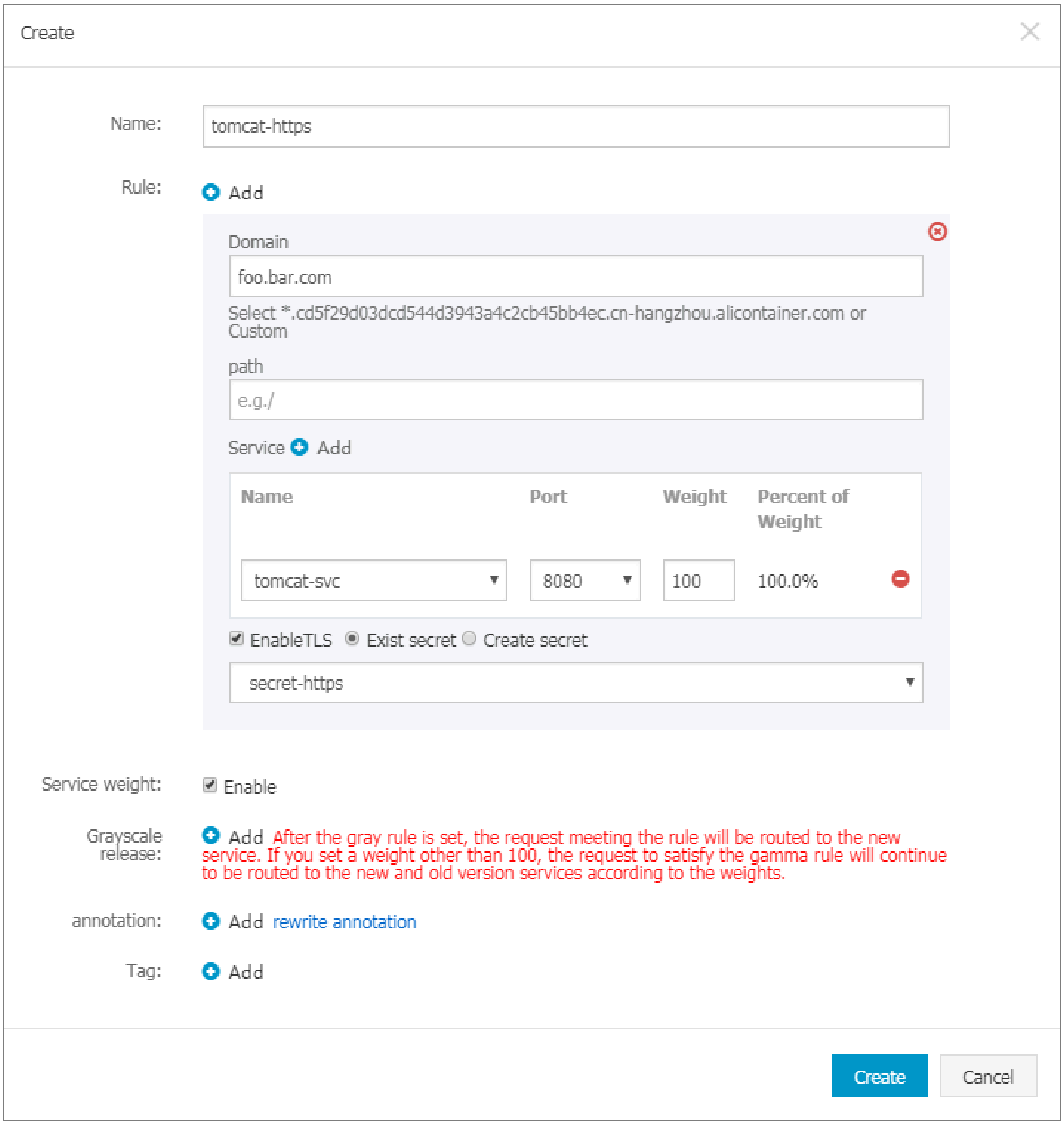
   Note:
   You must set the domain name properly. Otherwise, you will encounter exceptions when accessing the application through HTTPS.

   ```
   kubectl create secret tls secret-https --key tls.key --cert tls.crt
   ```

2. Log on to the *Container Service console*.
3. In the left-side navigation pane, 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 the Container Service console*. The configuration in this example is as follows:

   - Name: Enter an Ingress name.
   - Domain: Enter the domain name set in the preceding steps. It must be the same as that configured in the SSL certificate.
   - Service: Select the service corresponding to the tomcat application.The service port is 8080.
   - Enable TLS: After enabling TLS, select the existing secret.



You can also use a YAML file to create an Ingress. In this example, the YAML sample file is as follows:

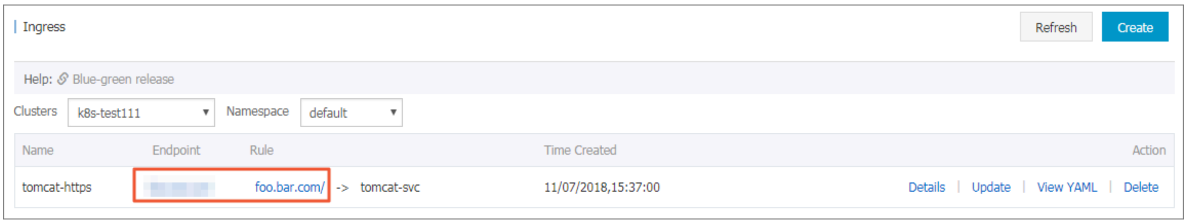
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
   name: tomcat-https
spec:
  tls:
  - hosts:
    - foo.bar.com
```

```
    secretName: secret-https
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /
        backend:
          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.

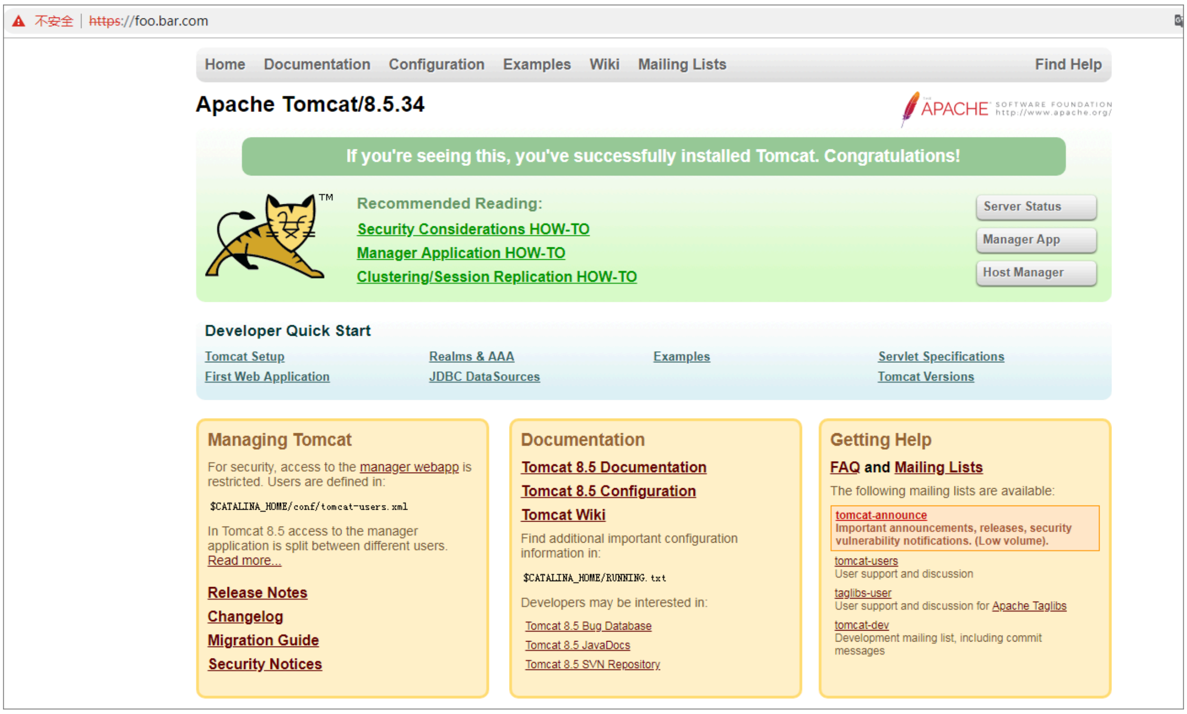
Note:
In this example, `foo.bar.com` is used as a testing domain name, and you need to create a record in the hosts file.

```
47.110.119.203   foo.bar.com                        #where, the IP
address is the Ingress endpoint.
```



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.



1.15 Release management

1.15.1 Manage a Helm-based release

Alibaba Cloud Container Service for Kubernetes is integrated with the package management tool Helm to help you quickly deploy applications on the cloud. However, Helm charts can be released multiple times and the release version must be managed. Container Service for Kubernetes provides a release function, which allows you to manage the applications released by using Helm in the Container Service console.

Prerequisites

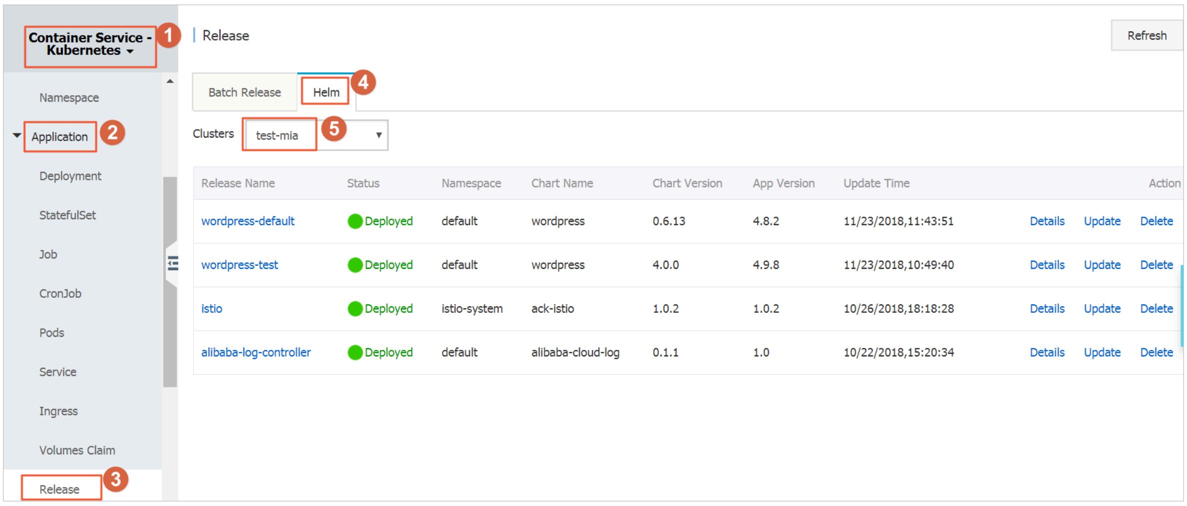
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have installed a Helm application by using the App Catalog function or Service Catalog function. For more information, see *Simplify Kubernetes application deployment by using Helm*. In this topic, the wordpress-default application is used as an example.

View release details

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

2. In the left-side navigation pane, select Container Service - Kubernetes. Then, select Application > Release and click the Helm tab. Select the target cluster from the Clusters drop-down list.

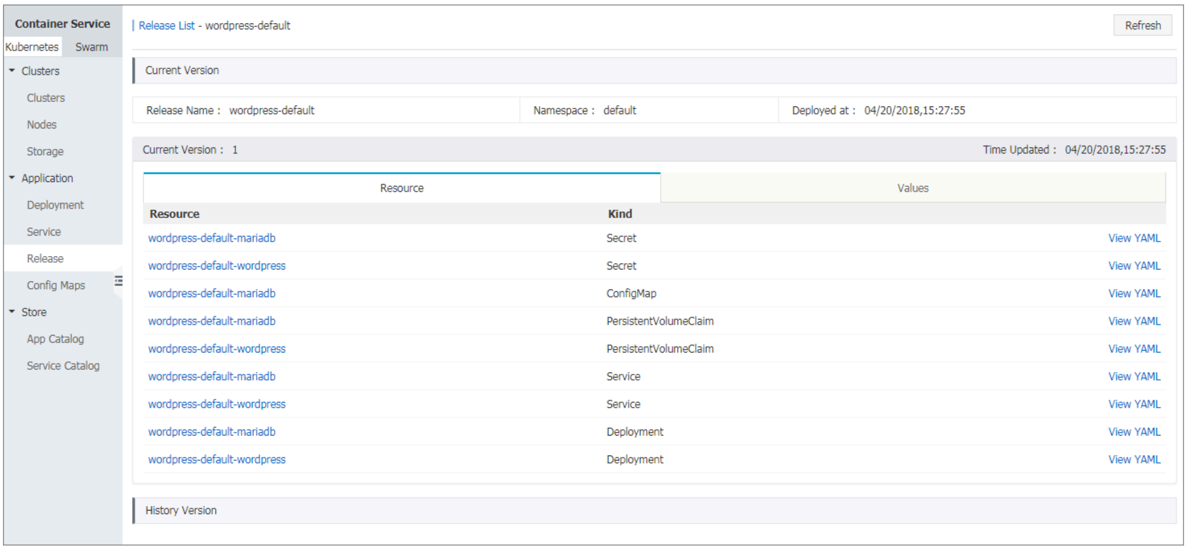
   In the displayed release list, you can view the applications and services released through Helm in the selected cluster.



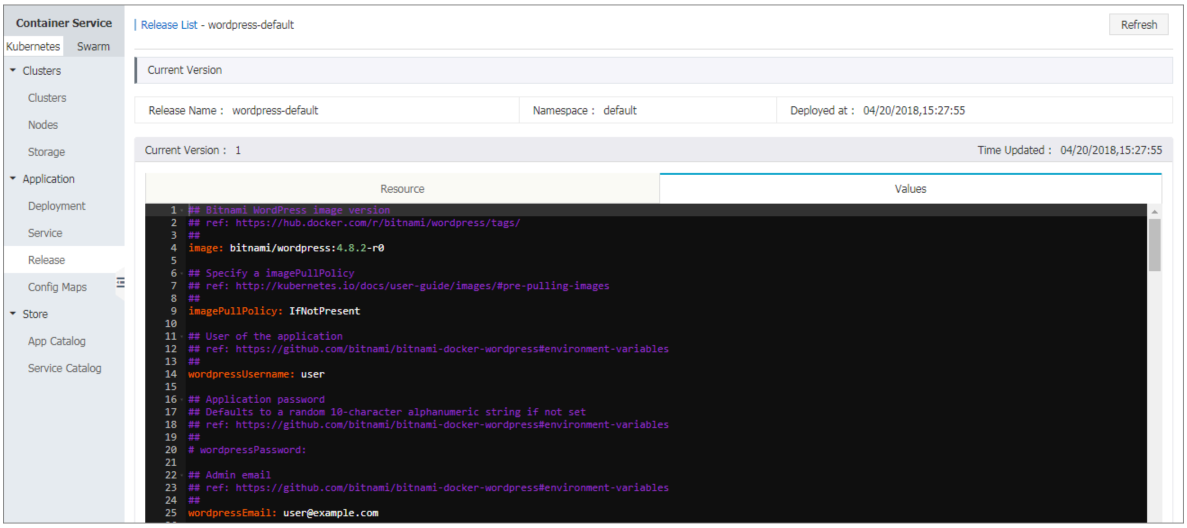
3. Find your target release (wordpress-default in this example) and click Details to view the release details.

   You can view such release details as the current version and history version. In this example, the current version is 1 and no history version exists. On the Resource tab page, you can view the resource information of wordpress-default, such as the resource name and the resource type, and view the YAML information.

   Note:
   You can view the running status of the resource in details by clicking the resource name and going to the Kubernetes dashboard page.



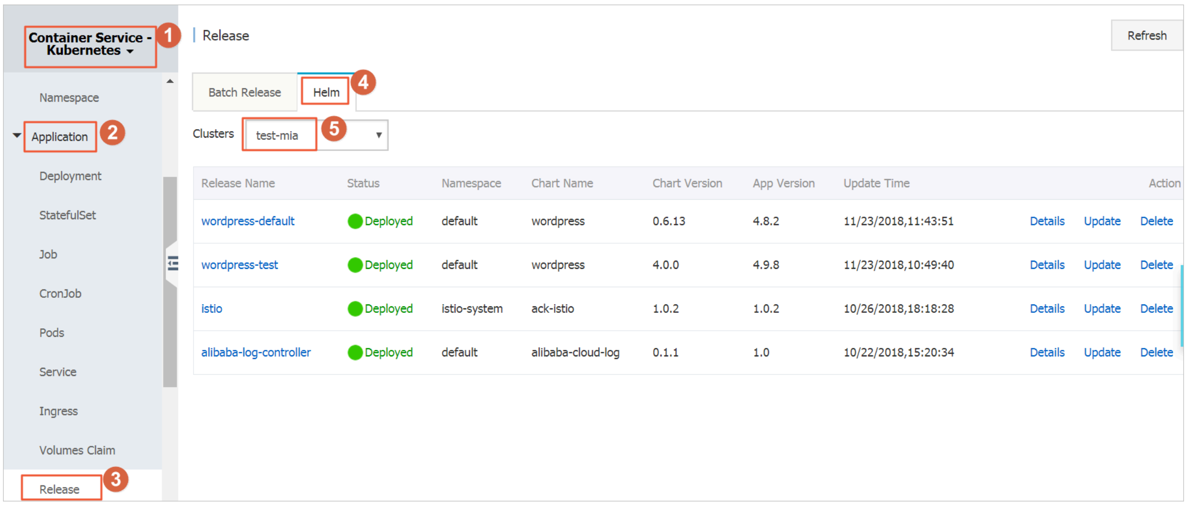
4. Click the Values tab to view the release parameters.



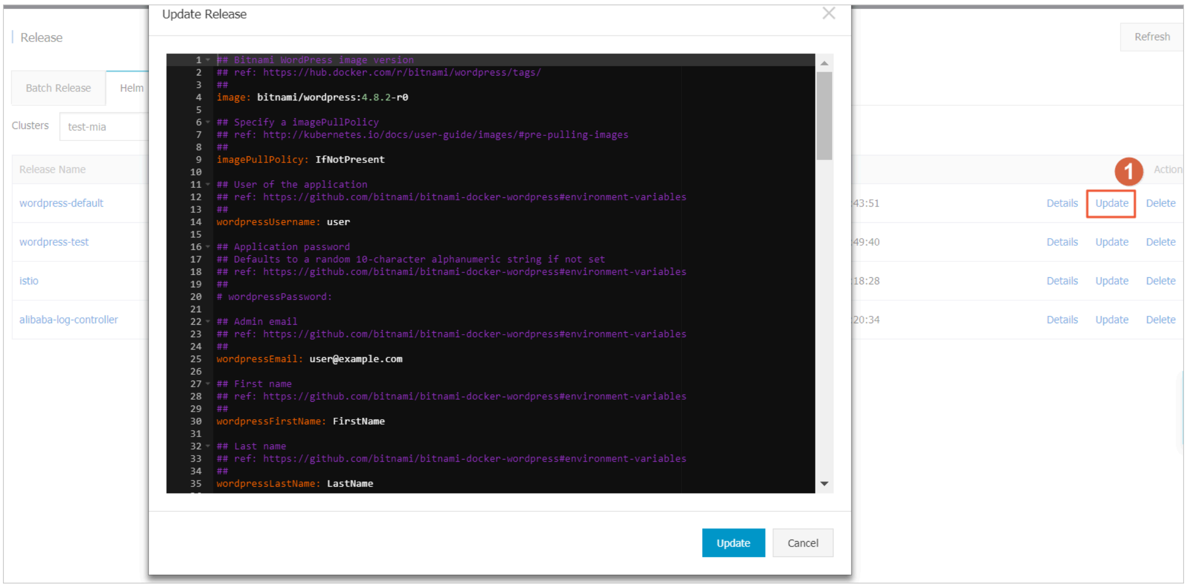
Update a release version

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, select Container Service - Kubernetes. Then, select Application > Release and click the Helm tab. Select the target cluster from the Clusters drop-down list.

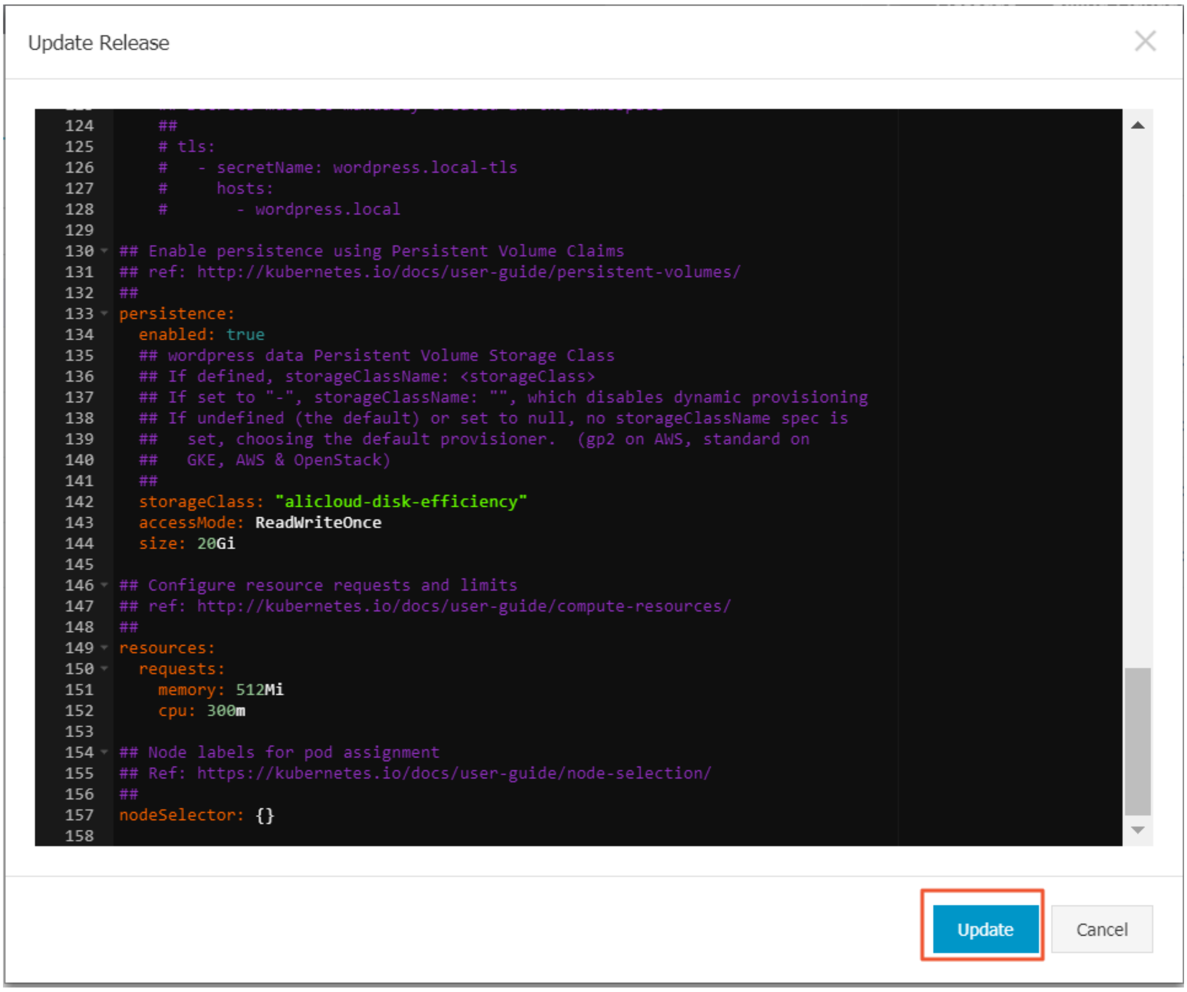
   In the displayed release list, you can view the applications and services released through Helm in the selected cluster.



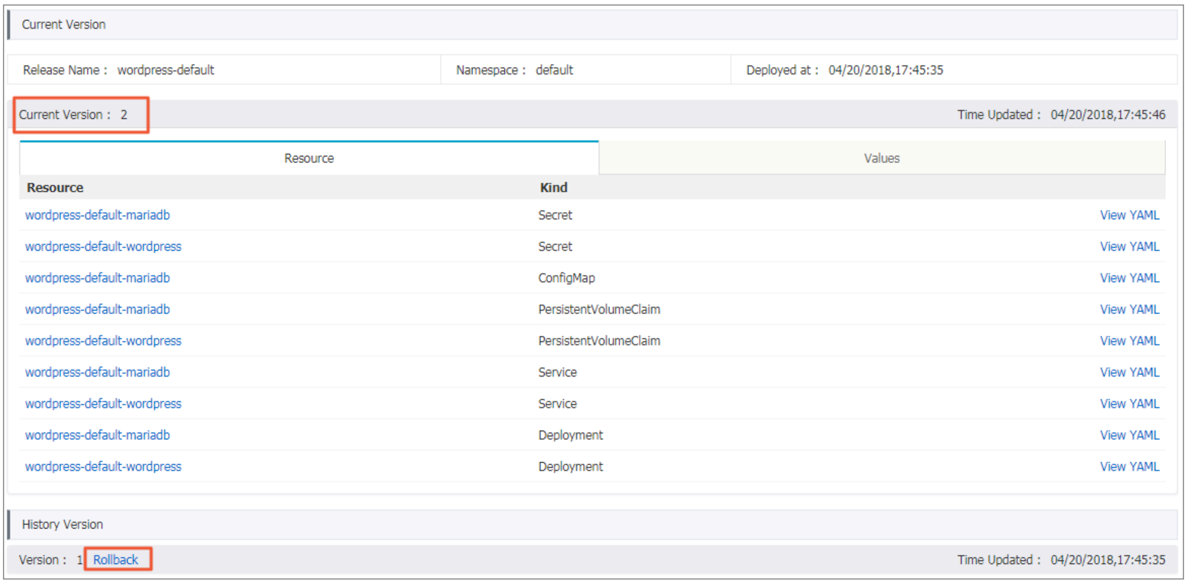
3. Find your target release (wordpress-default in this example). Click Update and the Update Release dialog box appears.



4. Modify the parameters and then click Update.



On the release list page, you can see that the current version changes to 2. To roll back to version 1, click Details and in the History Version area, click Rollback.

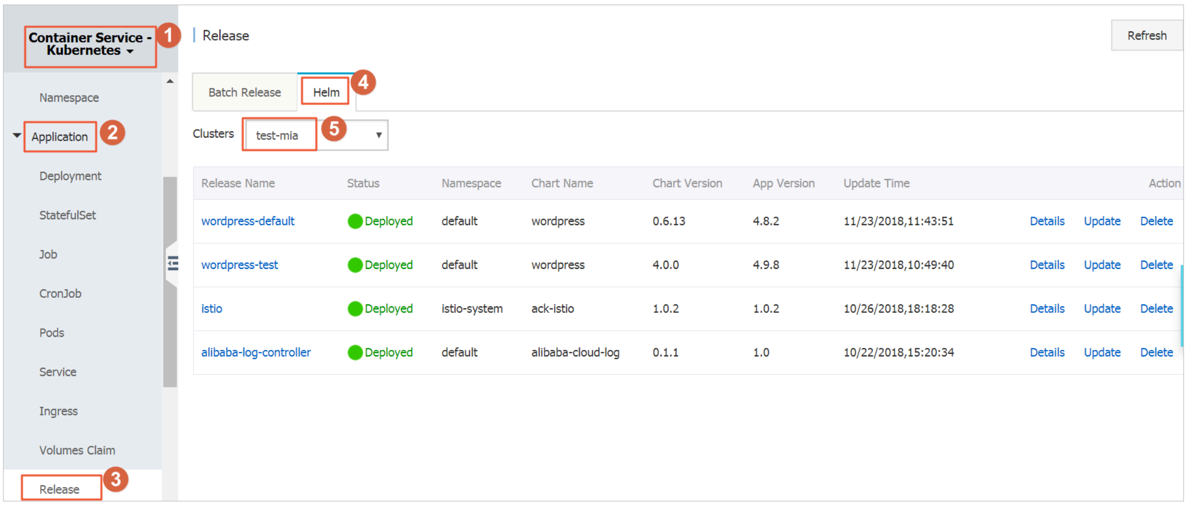


Delete a release

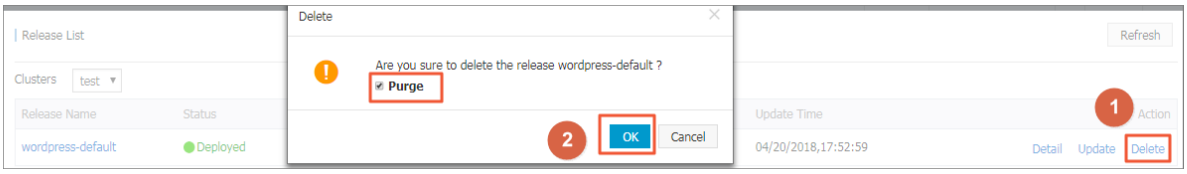
1. Log on to the *Container Service console*.

2. In the left-side navigation pane, select Container Service - Kubernetes. Then, select Application > Release and click the Helm tab. Select the target cluster from the Clusters drop-down list.

   In the displayed release list, you can view the applications and services released through Helm in the selected cluster.



3. Find your target release (wordpress-default in this example). Click Delete and the Delete dialog box appears.



4. Select the Purge check box if you want to clear the release records, and then click OK. After you delete a release, the related resources such as the services and deployments are deleted too.

1.15.2 Use batch release on Alibaba Cloud Container Service for Kubernetes

You can use Alibaba Cloud Container Service for Kubernetes to release application versions in batches, achieving fast version verification and rapid iteration of applications.

Context

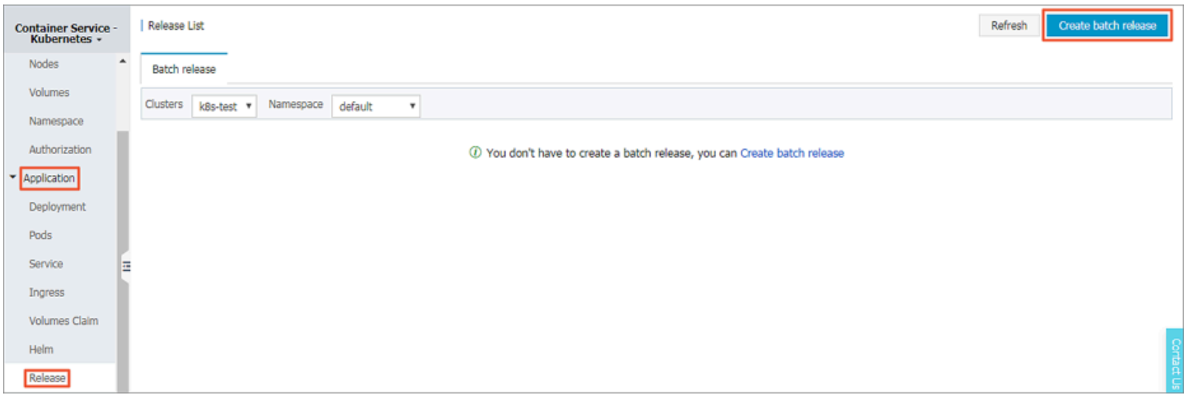
> **Note:**
> The latest Kubernetes cluster has installed alicloud-application-controller by default. For older versions of clusters, only versions of 1.9.3 and later are currently

supported, and you can upgrade old versions of clusters through the prompt link on the console.

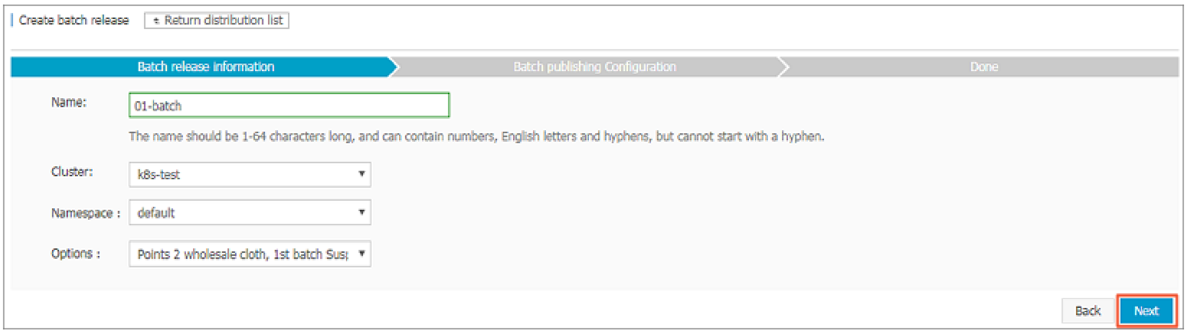
Procedure

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

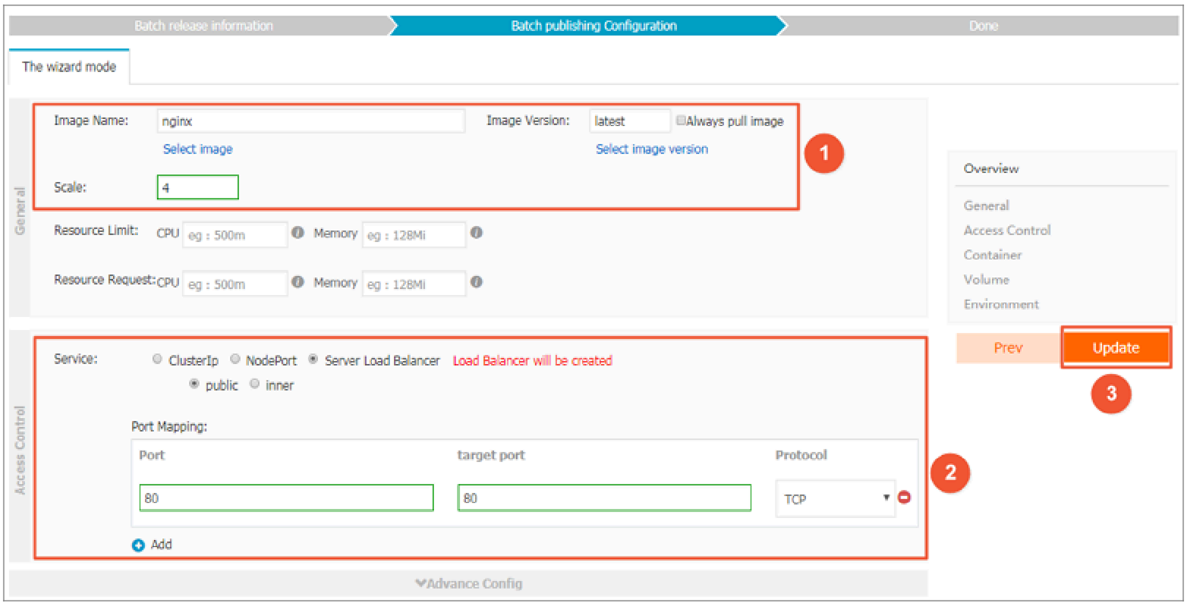
   Note:
   If the button is gray, you can upgrade the cluster by following the upgrade link.



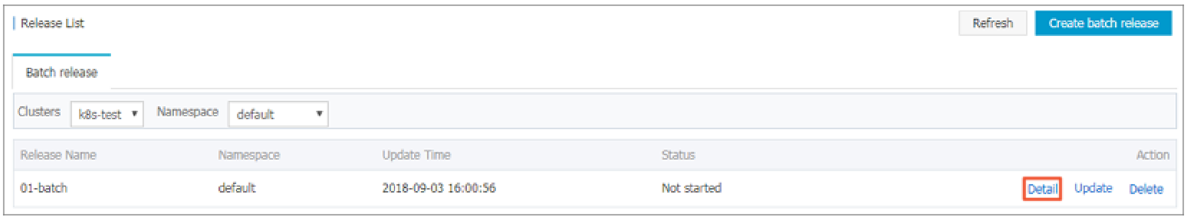
3. Configure batch release information, including the application name, cluster, namespace, and options. Click Next.



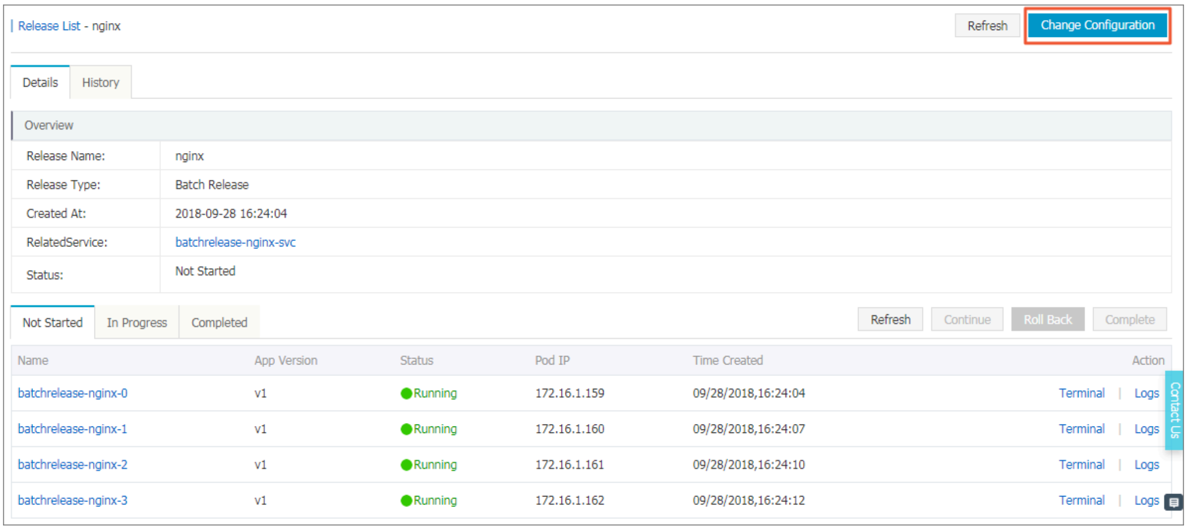
4. On the batch publishing configuration page, configure the backend pod and service, and then click Update to create an application.



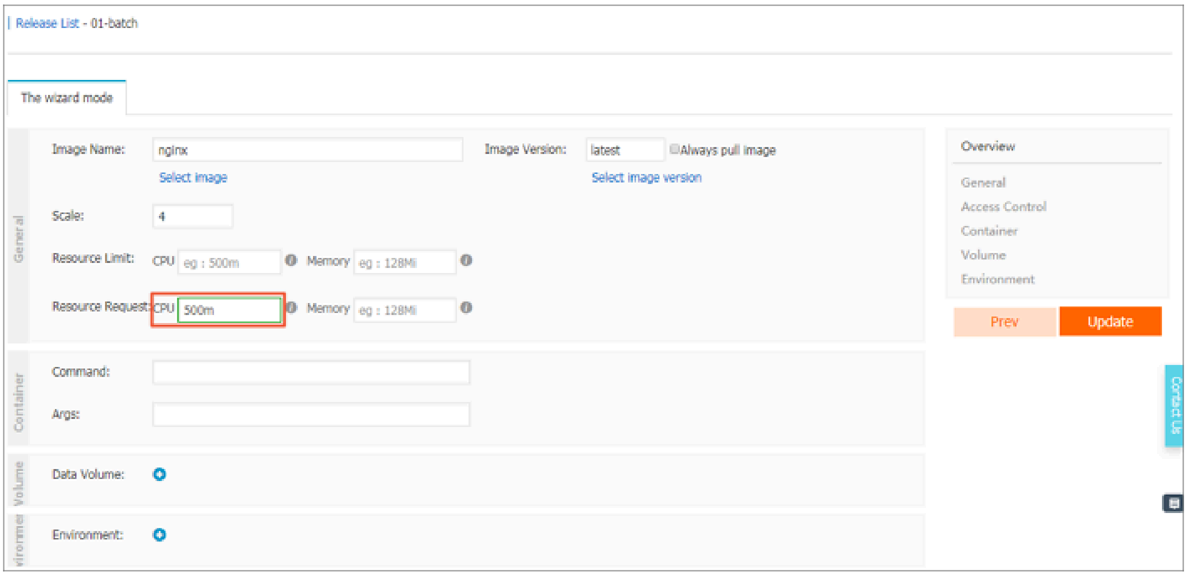
5. Return to the release list, an application is displayed in the Not started status. Click Detail on the right.



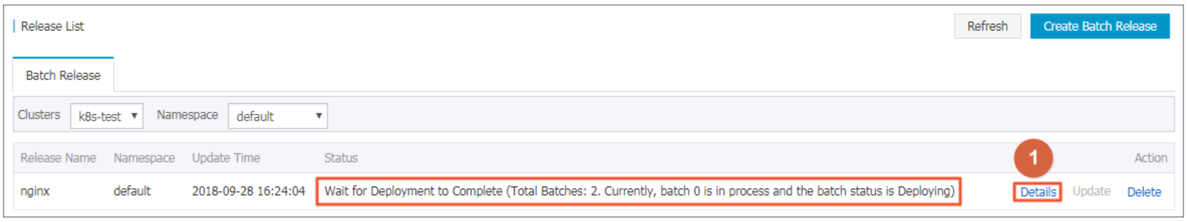
6. On the application detail page, you can view more information. Click Change Configuration in the upper-right corner of the page to make a batch release change.



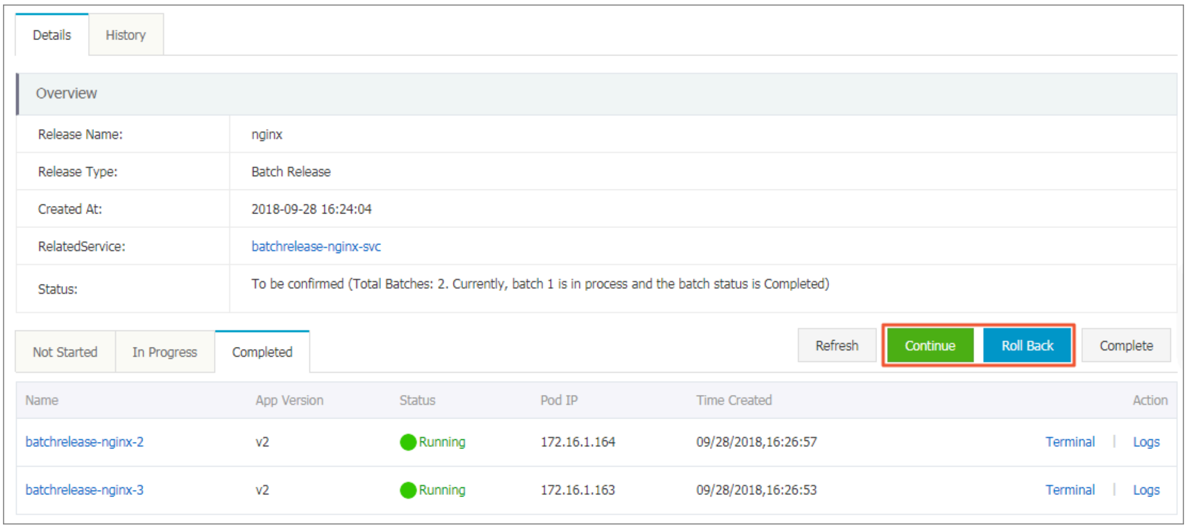
7. Configure changes for the new version of the application, and then click Update.



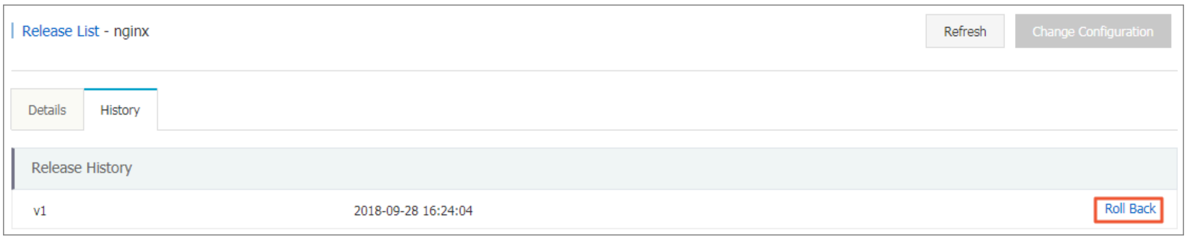
8. By default, you return to the release list page, where you can view the batch release status of the application. After completing the first deployment, click Detail.



9. You can see that the Not Started list is has two pods and the Completed list has two pods, which indicates that the first batch has been completed in batch release. Click Continue, you can release the second batch of pods. Click Roll Back to roll back to the previous version.



10.When completing the release, click Historyto roll back to history versions.



What's next

You can use batch release to quickly verify your application version without traffic consumption. Batch release is more resource-saving than blue-green release. Currently, batch release can be performed on only web pages. The yaml file editing is to be opened later to support more complex operations.

1.16 Istio management

1.16.1 Overview

Istio is an open platform that provides connection, protection, control and monitors microservices.

Microservices are currently being valued by more and more IT enterprises. Microservices are multiple services divided from a complicated application. Each service can be developed, deployed, and scaled. Combining the microservices and container technology simplifies the delivery of microservices and improves the liability and scalability of applications.

As microservices are extensively used, the distributed application architecture composed of microservices becomes more complicated in dimensions of operation and maintenance, debugging, and security management. Developers have to deal with greater challenges, such as service discovery, load balancing, failure recovery, metric collection and monitoring, A/B testing, gray release, blue-green release, traffic limiting, access control, and end-to-end authentication.

Istio emerged. Istio is an open platform for connecting, protecting, controlling, and monitoring microservices. It provides a simple way to create microservices networks and provides capabilities such as load balancing, inter-service authentication, and monitoring. Besides, Istio can provide the preceding functions without modifying services.

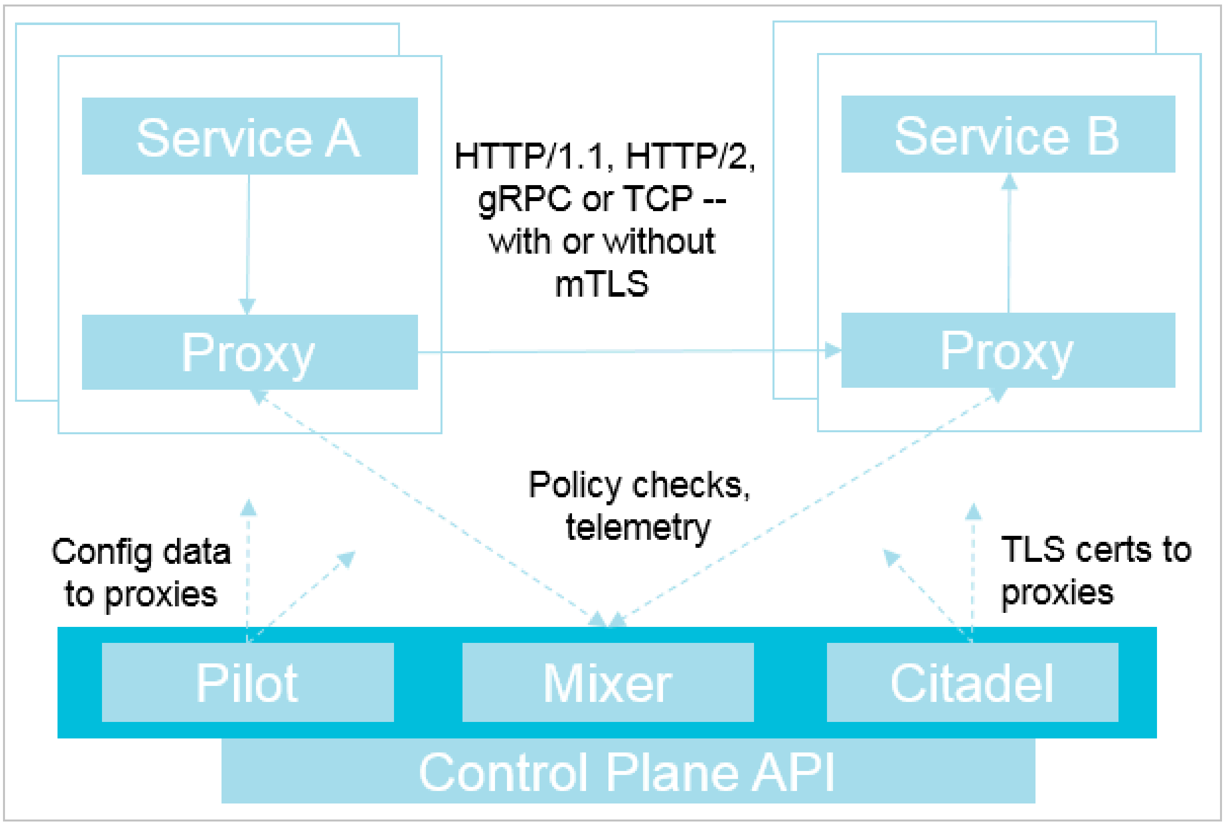
Istio provides the following functions:

- Traffic management: Controls traffic and API calls between services to enhance the system reliability.
- Authentication and security protection: Provides authentication for services in meshes, and protects the traffic of services to enhance the system security.
- Policy execution: Controls access policies between services without requiring changes to the services.
- Observability: Obtains traffic distribution and call relationships between services to quickly locate problems.

Istio architecture

Istio is logically divided into a control plane and a data plane:

- Control plane: Administration proxy (the default is Envoy) for managing traffic routing, runtime policy execution, and more
- Data plane: Consists of a series of proxys (the default is Envoy) for managing and controlling network communication between services.



Istio is composed of the following components:

- Istio Pilot: Collects and validates configurations, and propagates them to various Istio components. It extracts environment-specific implementation details from the policy execution module (Mixer) and the intelligent proxy (Envoy), providing them with an abstract representation of user services, independent of the underlying platform. In addition, traffic management rules (that is, generic Layer-4 rules and Layer-7 HTTP/gRPC routing rules) can be programmed through Pilot at runtime.

- Policy execution module (Mixer): Executes access control and usage policies across the service mesh, and collects telemetry data from the intelligent proxy (Envoy) and other services. Mixer executes policies based on the Attributes provided by the intelligent proxy (Envoy).
- Istio security module: Provides inter-service and inter-user authentication to guarantee enhanced security between services without modifying service codes. Includes three components:
  - Identification: When Istio runs on Kubernetes, it identifies the principal that runs the service according to the service account provided by container Kubernetes.
  - Key management: Provides CA automated generation, and manages keys and certificates.
  - Communication security: Provides a tunnel between the client and the server through the intelligent proxy (Envoy) to secure services.
- Intelligent proxy (Envoy): Deployed as an independent component in the same Kubernetes pod along with relevant microservice, and provides a series of attributes to the policy execution module (Mixer). The policy execution module (Mixer) uses these attributes as the basis to execute policies, and sends them to monitoring systems.

1.16.2 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 *Kubernetes permission configuration guide for RAM users*.

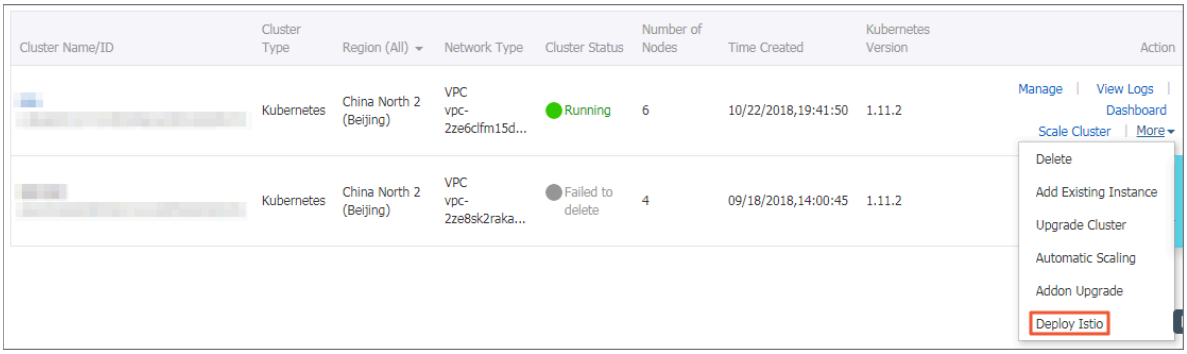
Background information

- 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.



4. Deploy Istio according to the following information.

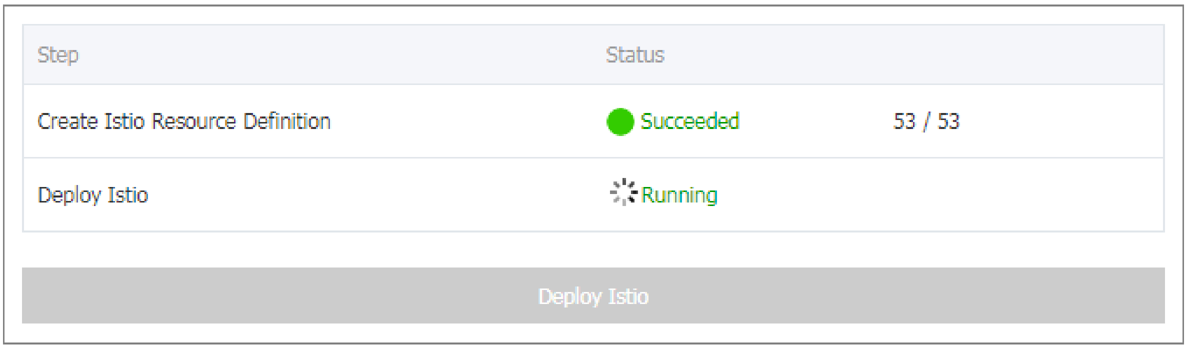
Configuration	Description
Clusters	Target cluster in which Istio is deployed.
Namespace	Namespace in which Istio is deployed.
Release Name	Name of Istio to be released.

Configuration	Description
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. · 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.

Configuration	Description
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: 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.



Expected results:

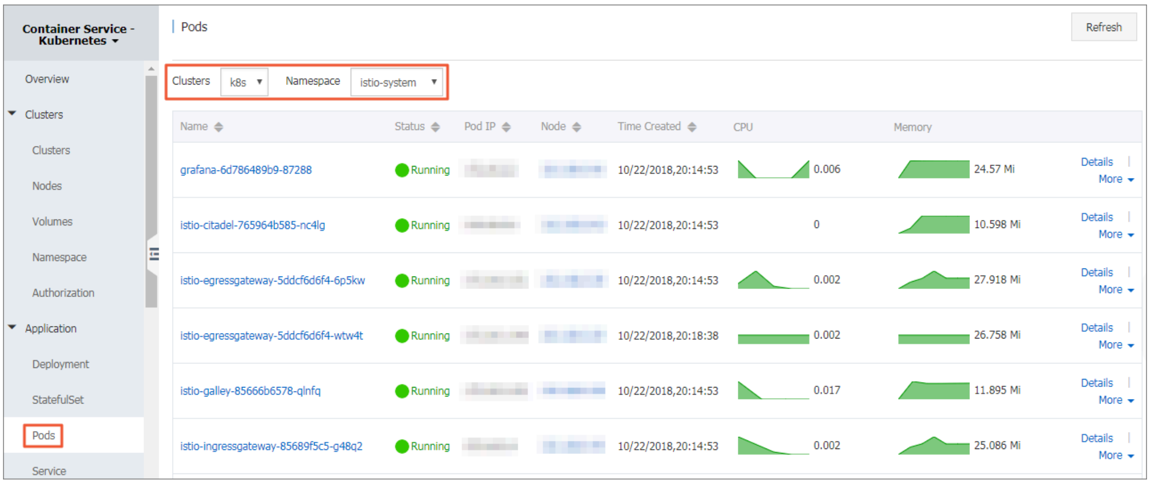
You can view your deployment results in the following ways:

- At the bottom of the Deploy Istio page, Deploy Istio is changed to Deployed.

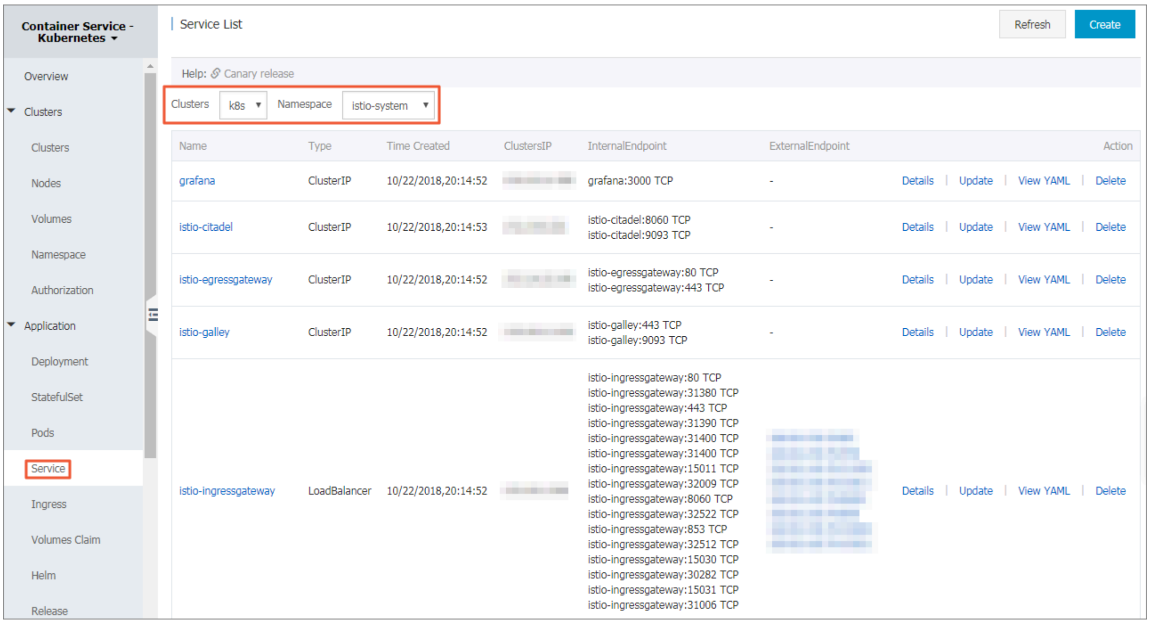
Step	Status
Create Istio Resource Definition	pending
Deploy Istio	pending

Deployed

- - 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.

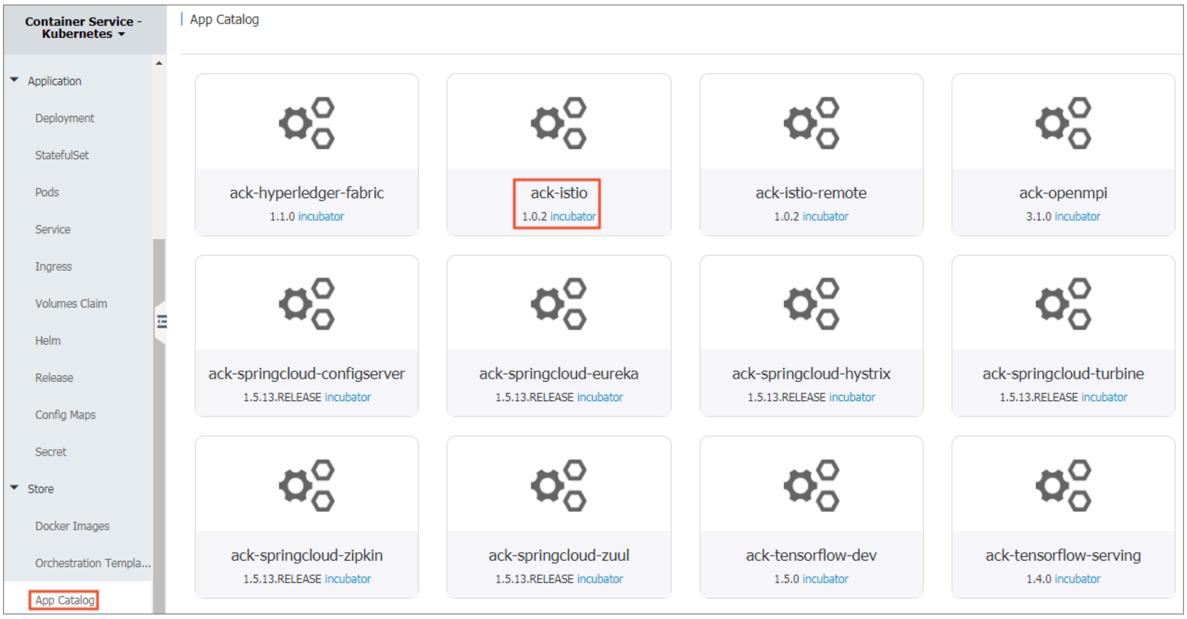


- - In the left-side navigation pane, click Application > Service.
  - Select the cluster and namespace in which Istio is deployed, and you can see the access addresses provided by the relevant services in which Istio is deployed.

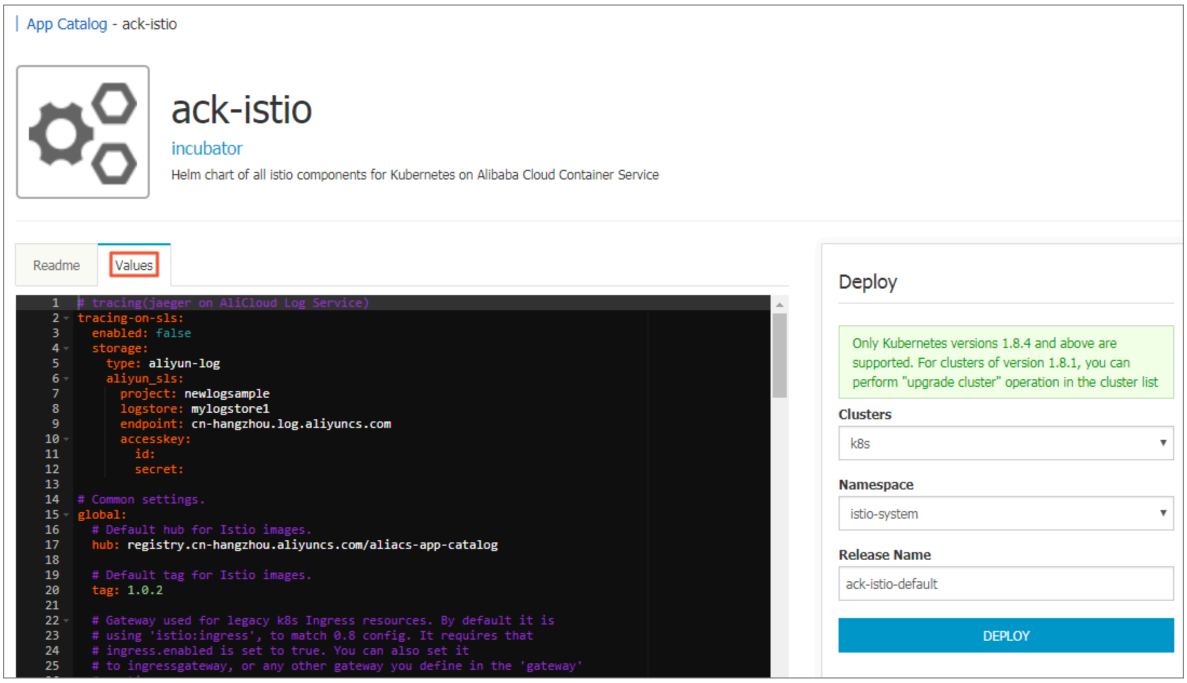


Use the application catalog to deploy Istio

1. Log on to the *Container Service console*.
2. In the left-side navigation pane, click Store > App Catalog.



3. Click ack-istio.
4. Click the Values tab to configure parameters.



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
```

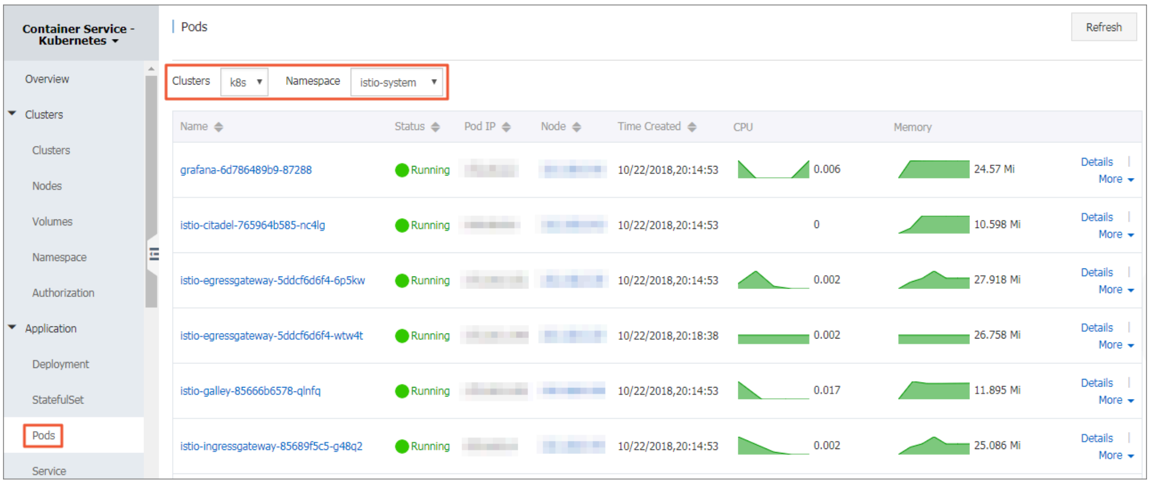
5. In the Deploy section on the right, configure the following.

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.

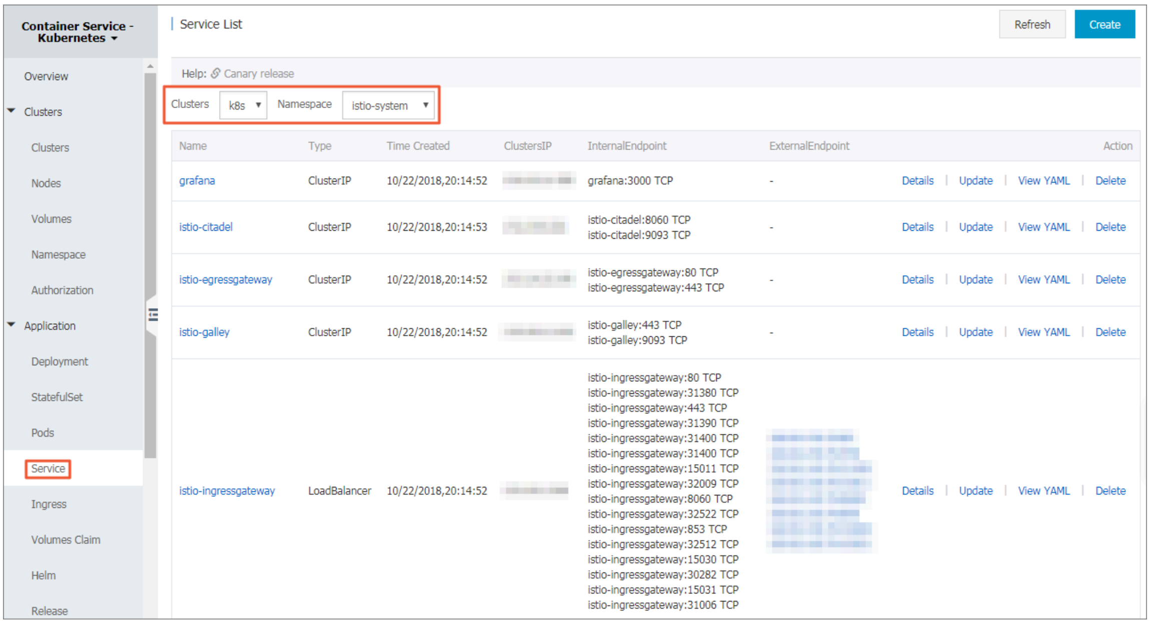
6. Click DEPLOY to start deployment.

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.



- - 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.



1.16.3 Update Istio

You can modify the deployed Istio through updates.

Prerequisites

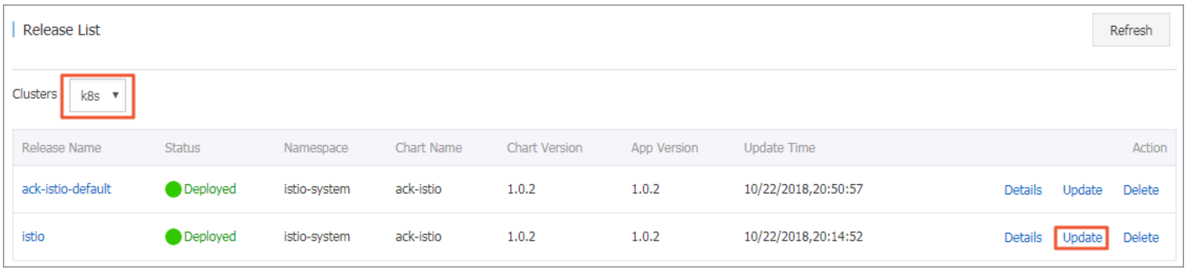
- You have created an Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created an Istio. For more information, see *Deploy Istio*.

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.

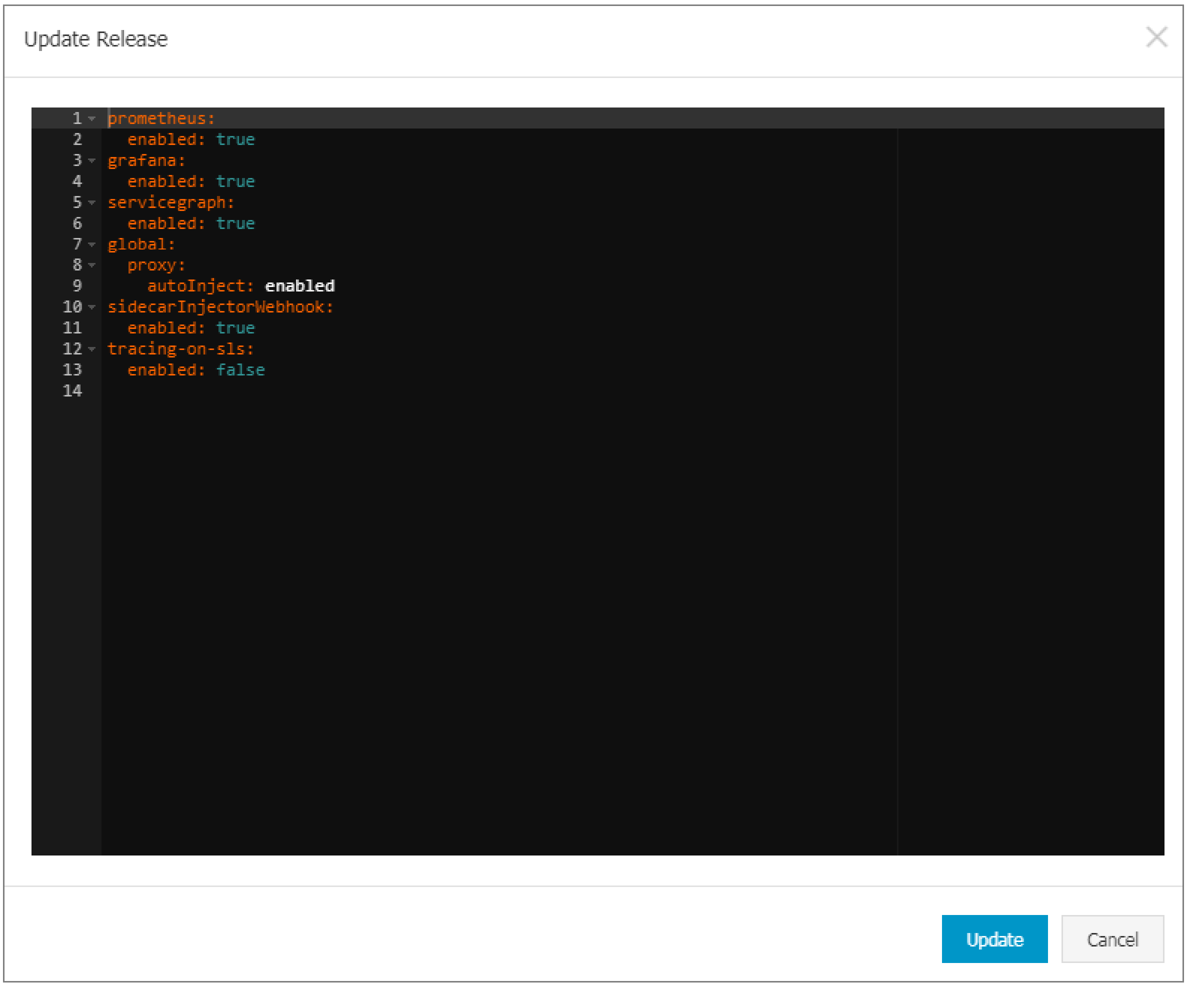
Note:

- 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.



4. In the displayed dialog box, modify parameters of the Istio, and then click Update.

   In this example, update the Istio that is deployed through the cluster interface:



Result

You can view updated content in two ways:

- After you complete the update, the page automatically jumps to theRelease List page. On the Resource tab, you can view updated content.
- Under the Kubernetes menu, click Application > Pods, and select the target cluster and namespace to view updating results.

1.16.4 Delete Istio

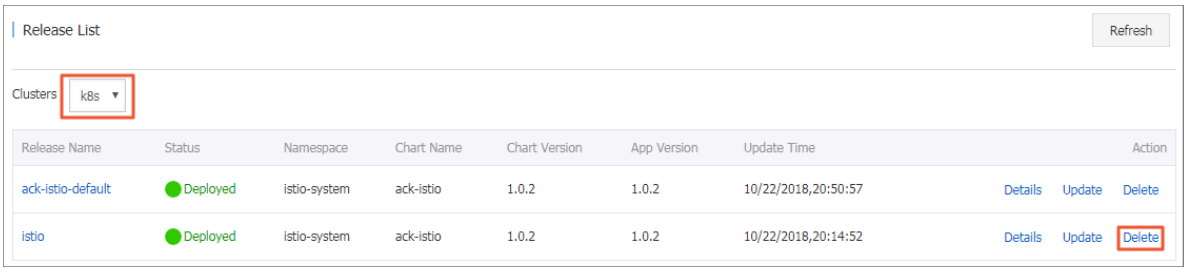
You can delete a deployed Istio through the deletingoperation.

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

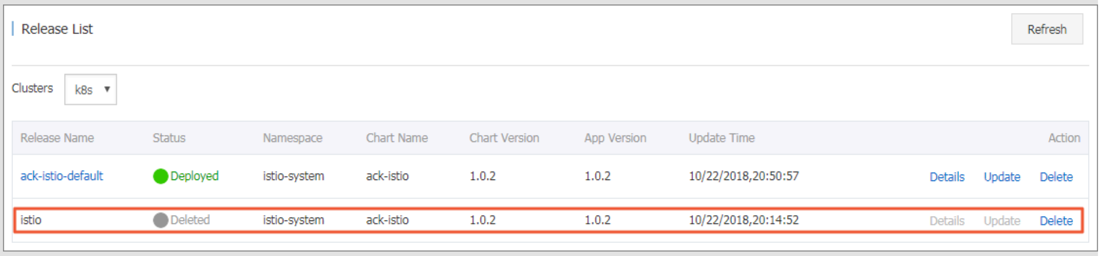
1. Log on to the *Container Service console*.
2. Under the Kubernetes menu, click Application > Helm in the left-side navigation pane.
3. Select a cluster, select the Istio to be deleted, and click Delete in the action column.



4. In the displayed dialog box, click OK.

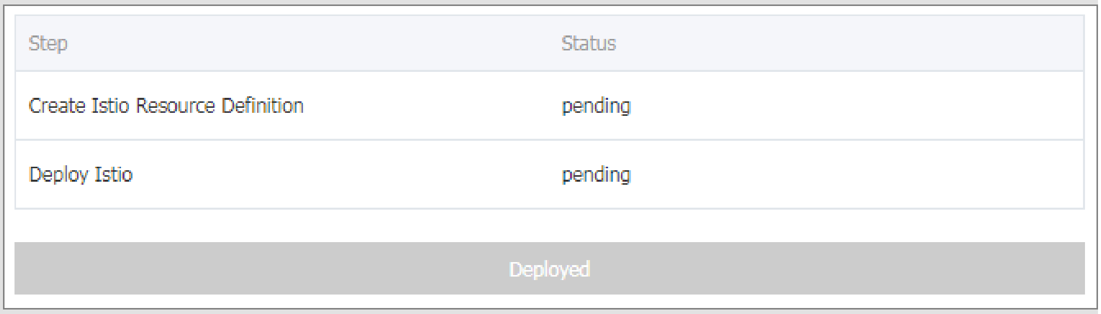
Note:

- Do not select the Purge box:
  - Releasing records are not deleted:

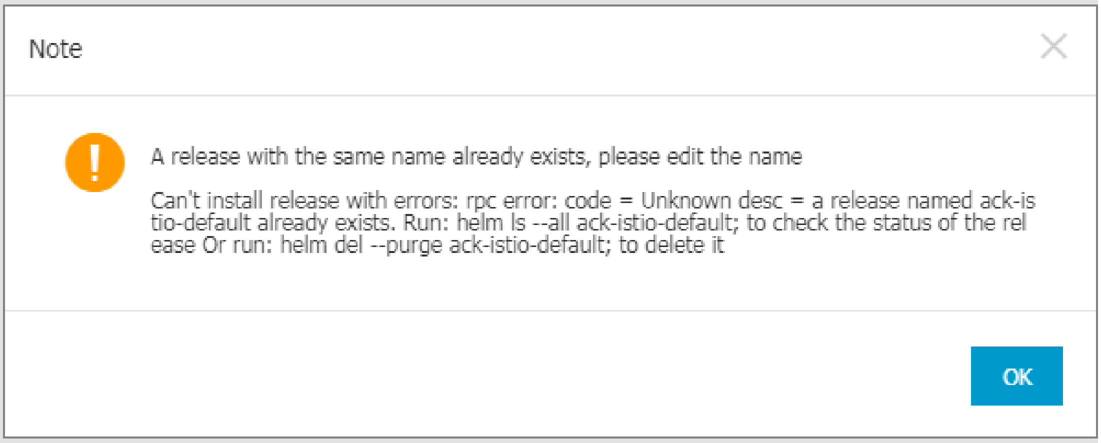


- The name of this Istio cannot be used again.

  When you redeploy the Istio through the cluster interface, the deployment status is deployed.



  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.



- 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.

1.16.5 Upgrade Istio components

This topic describes how to upgrade Istio components.

Background information

- The Istio upgrade may install new binaries, and change configurations and API schemas.
- The upgrade process may cause service downtime.
- To minimize downtime, use multiple replicas to ensure that your Istio control plane components and your applications remain highly available.

Note:

In the following example, assume that the Istio components are installed and upgraded in the istio-system namespace.

Download the installation package

Alibaba Cloud provides the installation file of the latest version, including Helm Chart. To download the latest installation package (v1.0.5), open the following download address:

```
https://aliacs-app-catalog.oss-cn-hangzhou.aliyuncs.com/charts-
incubator/ack-istio-1.0.5.tgz
```

After you download the file, extract it to the *{ack-istio}* directory. The following file structure is displayed:

```
.
├── Chart.yaml
├── LICENSE
├── README.md
├── charts
│   ├── certmanager
│   ├── galley
│   ├── gateways
│   ├── grafana
│   ├── ingress
│   ├── kiali
│   ├── mixer
│   ├── pilot
│   ├── prometheus
│   ├── security
│   ├── servicegraph
│   ├── sidecarInjectorWebhook
│   └── tracing-on-sls
├── requirements.lock
├── requirements.yaml
├── templates
│   ├── _affinity.tpl
│   ├── _helpers.tpl
│   ├── configmap.yaml
```

```
│    ├── crds.yaml
│    ├── install-custom-resources.sh.tpl
│    └── sidecar-injector-configmap.yaml
└── values.yaml
```

Procedure

To complete the upgrade process, you need to upgrade CRD files, the control plane, and the data plane sidecar.

Upgrade CRD files

1. In the *{ack-istio}* directory, you can view CRDs files.
2. Run the following commands to upgrade the CRD files of Istio:

```
kubectl apply -f {ack-istio}/templates/crds.yaml -n istio-system
```

```
kubectl apply -f {ack-istio}/charts/certmanager/templates/crds.yaml
 -n istio-system
```

Upgrade the control plane

The Istio control plane components include the Citadel, Pilot, Policy, Telemetry and Sidecar injector.

1. Run the following command to view the parameter configurations of the deployed Istio:

```
helm get values istio > current.values.yaml
```

2. Merge the content of the *current.values.yaml* file to the *values.yaml* file in the ack-istio directory. You must ensure that the version of *values.yaml* is the same as that of the new Istio installation file. The following is the setting of this example:

```
global:
  tag: 1.0.5
```

3. Before upgrading, run the following command to perform a dry-run check:

```
helm upgrade --dry-run --debug istio {ack-istio} --namespace istio-
system  -f values.yaml
```

4. In the left-side navigation pane of the Container Service console, choose Application > Release.
5. Select the target cluster, select the target Release Name, and click Upgrade in the Action column.
6. Replace the content of the displayed dialog box with the content of the updated *values.yaml* file. Then, click Update.

Release Name	Status	Namespace	Chart Name	Chart Version	App Version	Update Time	Action
istio	Deployed	istio-system	ack-istio	1.0.5	1.0.5	12/27/2018,09:51:14	Details Update Delete

Upgrade the data plane sidecar

Note that after you upgrade the control plane, the applications that have already run Istio will still use the sidecar of an earlier version. To upgrade the sidecar, you need to re-inject it.

Automatic sidecar injection

If you use automatic sidecar injection, you can upgrade the sidecar by performing a rolling update for all pods. Then, the sidecar of the new version will be automatically re-injected.

You can use the following script to trigger the rolling update by patching the termination grace period.

```
NAMESPACE=$1
DEPLOYMENT_LIST=$(kubectl -n $NAMESPACE get deployment -o jsonpath='{.
items[*].metadata.name}')
echo "Refreshing pods in all Deployments: $DEPLOYMENT_LIST"
for deployment_name in $DEPLOYMENT_LIST ; do
    #echo "get TERMINATION_GRACE_PERIOD_SECONDS from deployment: $
deployment_name"
    TERMINATION_GRACE_PERIOD_SECONDS=$(kubectl -n $NAMESPACE get
 deployment "$deployment_name" -o jsonpath='{.spec.template.spec.
terminationGracePeriodSeconds}')
    if [ "$TERMINATION_GRACE_PERIOD_SECONDS" -eq 30 ]; then
        TERMINATION_GRACE_PERIOD_SECONDS='31'
    else
        TERMINATION_GRACE_PERIOD_SECONDS='30'
    fi
    patch_string="{\"spec\":{\"template\":{\"spec\":{\"terminatio
nGracePeriodSeconds\":$TERMINATION_GRACE_PERIOD_SECONDS}}}}"
    #echo $patch_string
    kubectl -n $NAMESPACE patch deployment $deployment_name -p $
patch_string
done
echo "done."
```

Manual sidecar injection

Run the following command to manually upgrade the sidecar:

```
kubectl apply -f <(istioctl kube-inject -f $ORIGINAL_DEPLOYMENT_YAML)
```

If the sidecar was previously injected with some customized injection configuration files, run the following command to manually upgrade the sidecar:

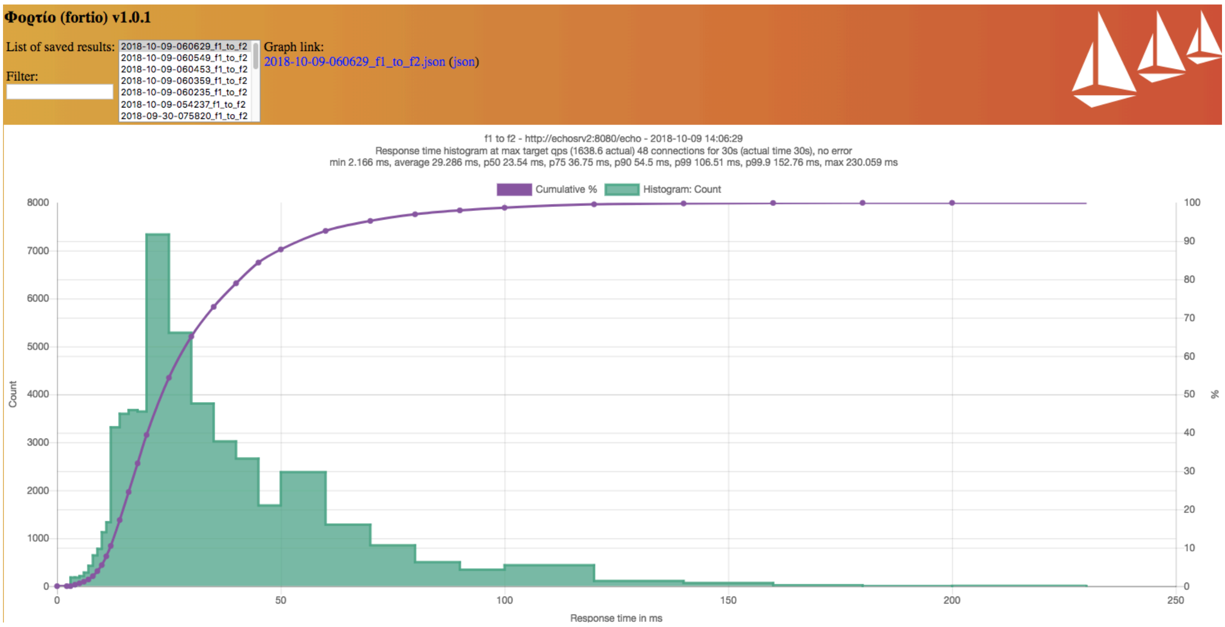
```
kubectl apply -f <(istioctl kube-inject --injectConfigFile inject-
config.yaml --filename $ORIGINAL_DEPLOYMENT_YAML)
```

Impacts caused by the Istio upgrade

Impacts caused by the CRD file upgrade

The upgrade process does not impact the calls between services within the cluster or the calls from the gateway to services.

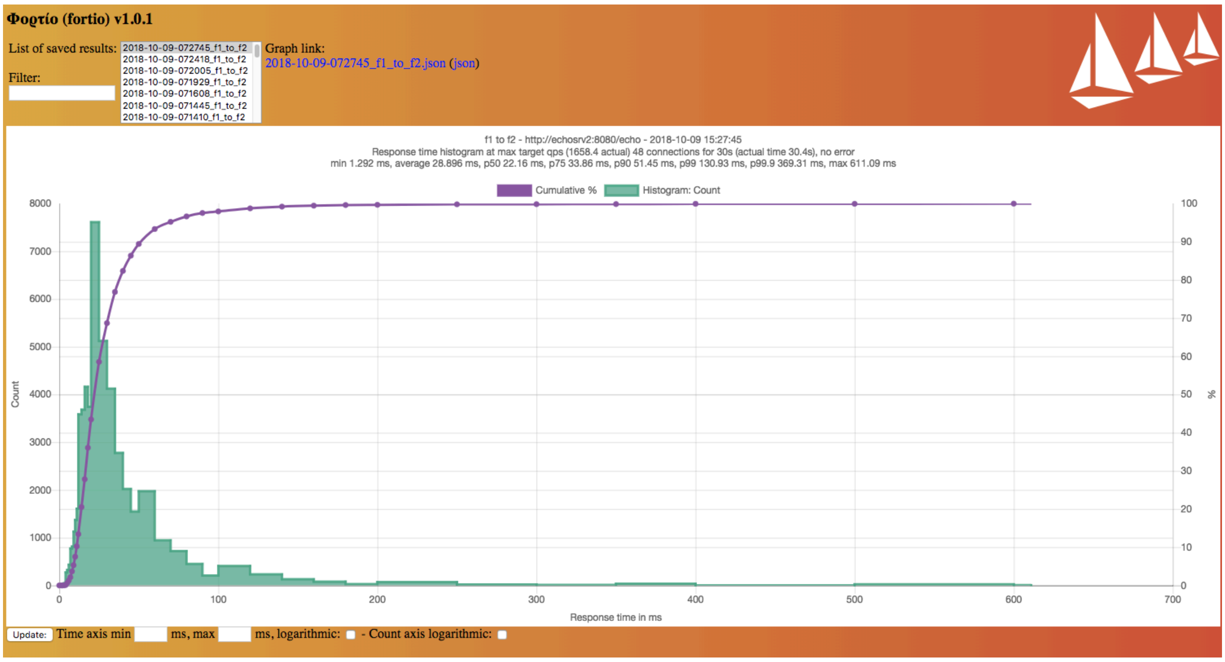
Calls between services within the cluster.



Impacts caused by the control plane upgrade

If HA is enabled, that is, the replicas of Pilot is 2, the HPA setting of `istio-pilot/istio-policy/istio-telemetry` is `minReplicas: 2`.

If you have changed the Istio version multiple times by upgrading or rolling back the component version, testing results will indicate that the QPS of calls between services remains unchanged and the calls proceed normally.



Impacts caused by the control plane sidecar upgrade

No obvious change occurs to both the QPS of the calls between services within the cluster and the QPS of the calls from the gateway to services. But these calls will terminate temporarily. We recommend that you use multiple replicas to upgrade the sidecar to reduce the impacts.

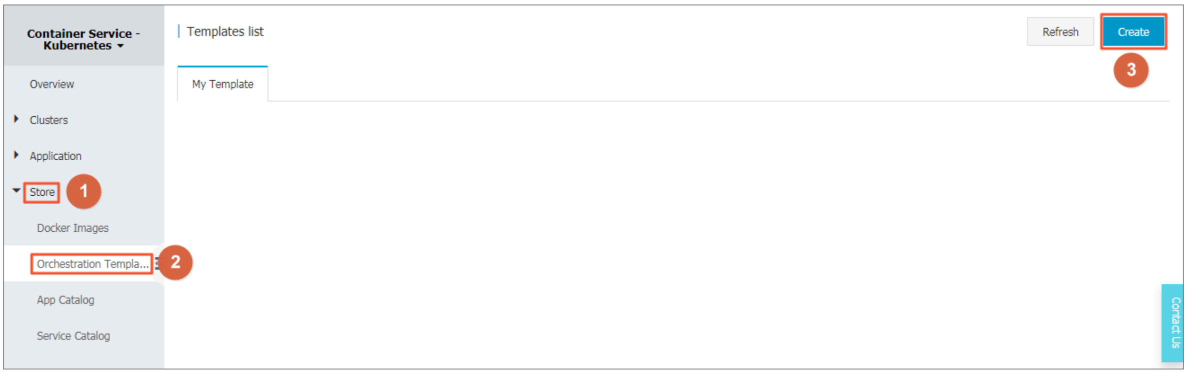
1.17 Template management

1.17.1 Create an orchestration template

You can use multiple methods to create orchestration templates through the Container Service console.

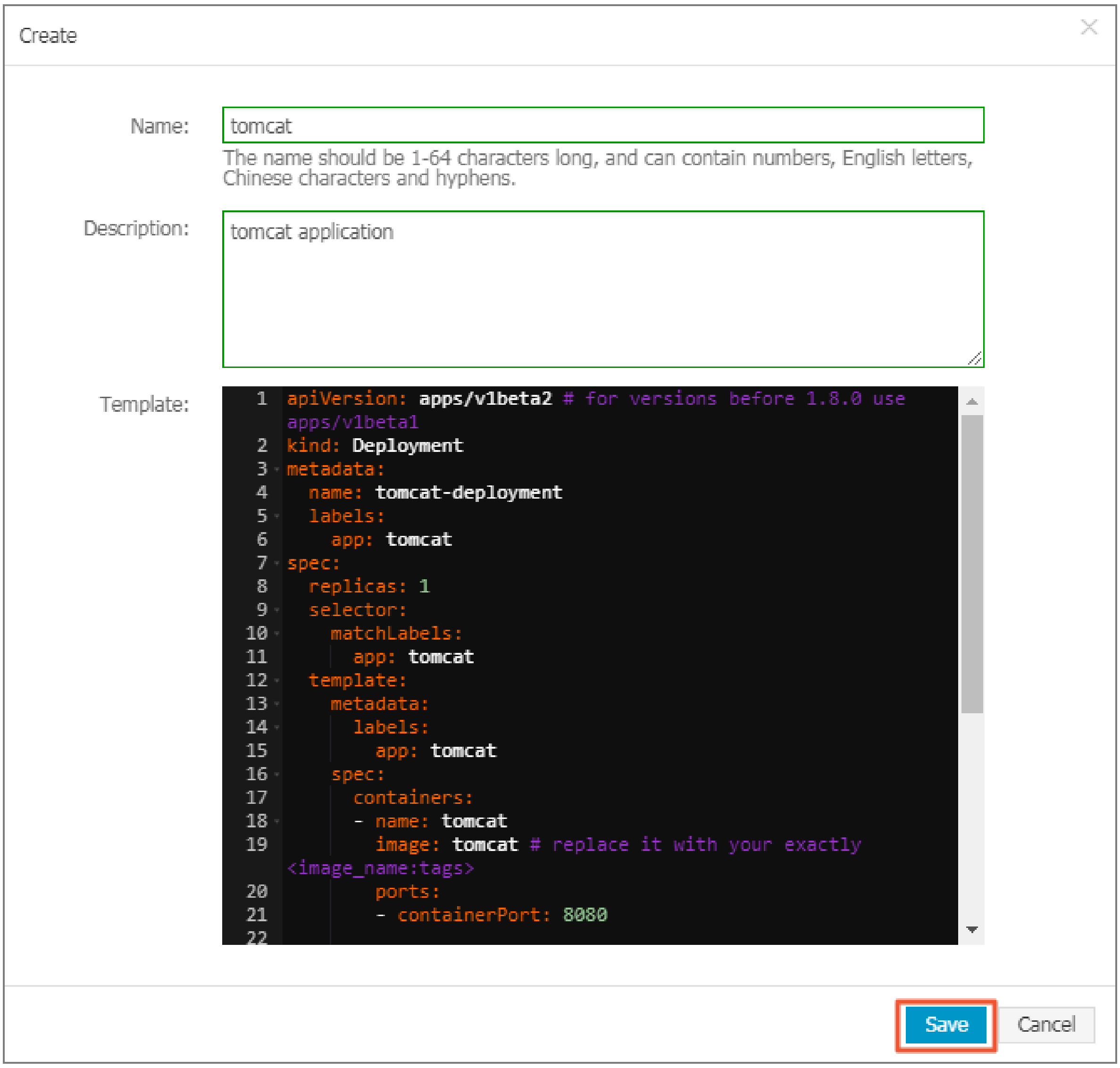
Procedure

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.

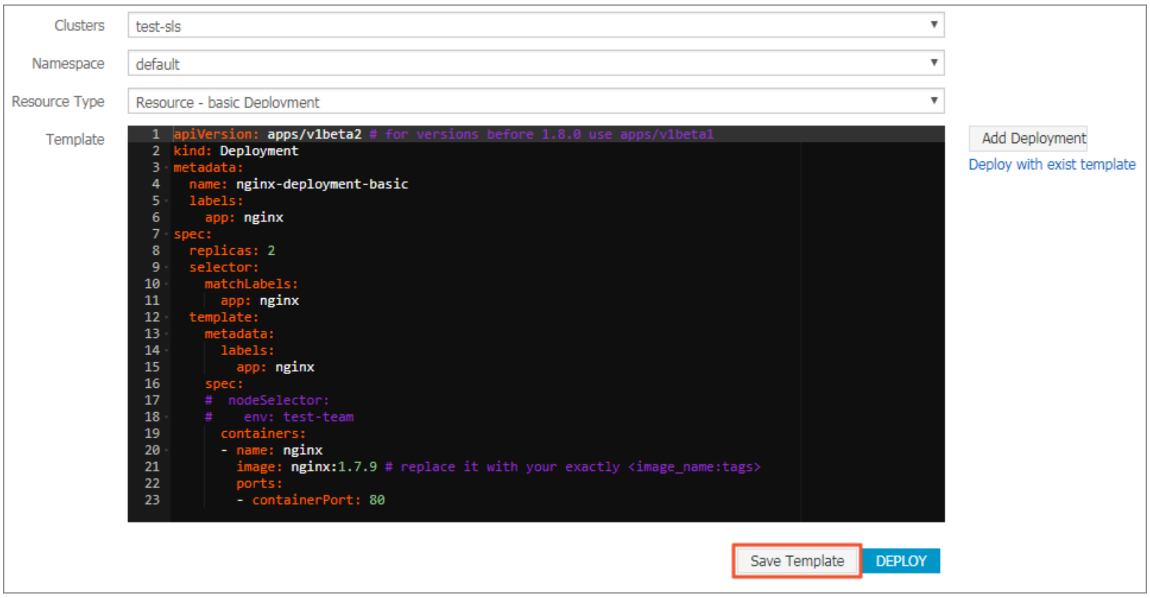
   - Name: Set the template name.
   - Description: Enter the description for the template. This parameter is optional.
   - Template: Configure the template that conforms to Kubernetes yaml syntax rules. The template can contain multiple resource objects that are separated by `---`.



4. After the template is created, the Template List page is displayed. You can see the template under My template.

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.

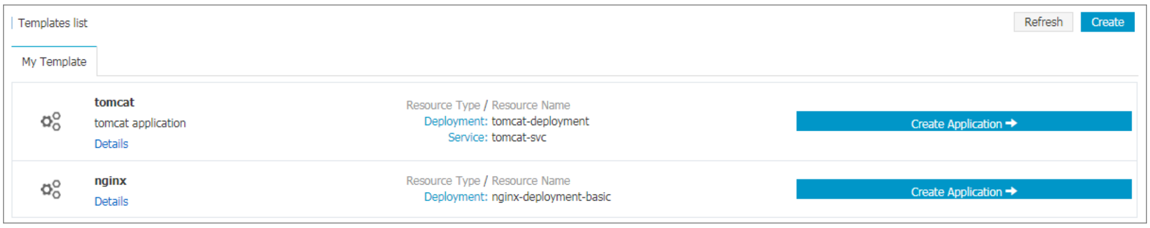
   a) Select a built-in template and click Save Template.



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.



What's next

You can quickly create an application by using the orchestration template under My Template.

1.17.2 Edit an orchestration template

You can edit an orchestration arrangement template.

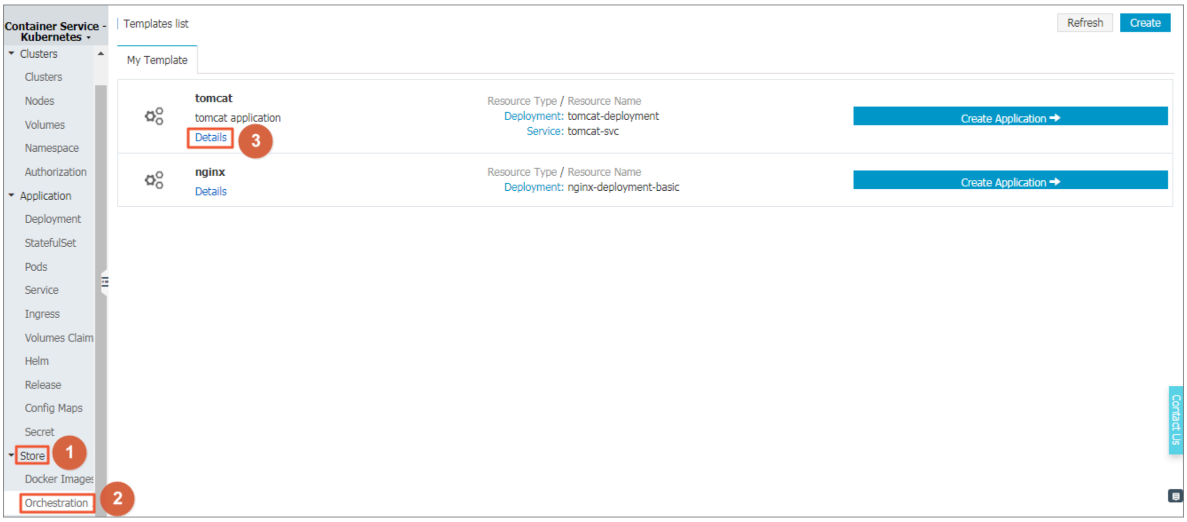
Prerequisites

You have created an orchestration template, see *Create an orchestration template*.

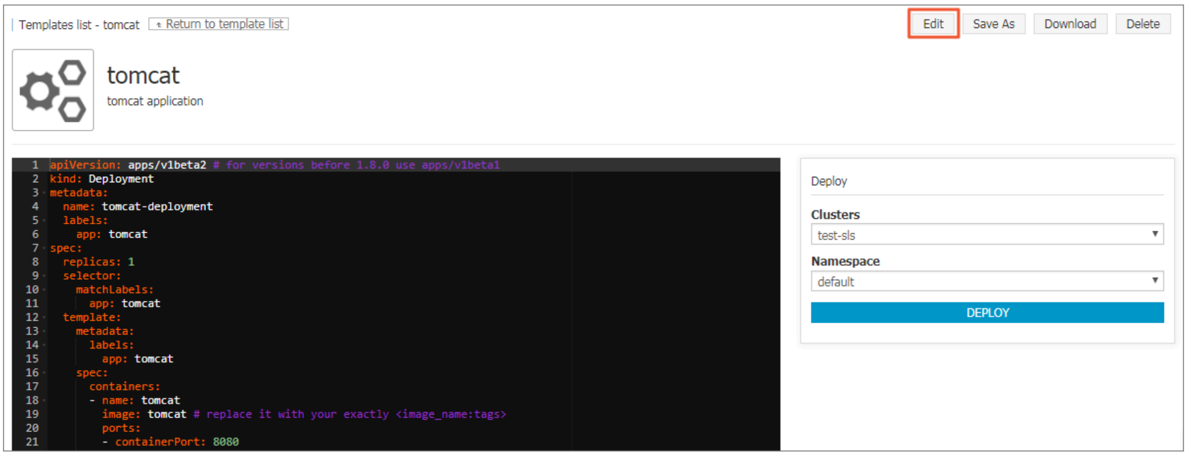
Procedure

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

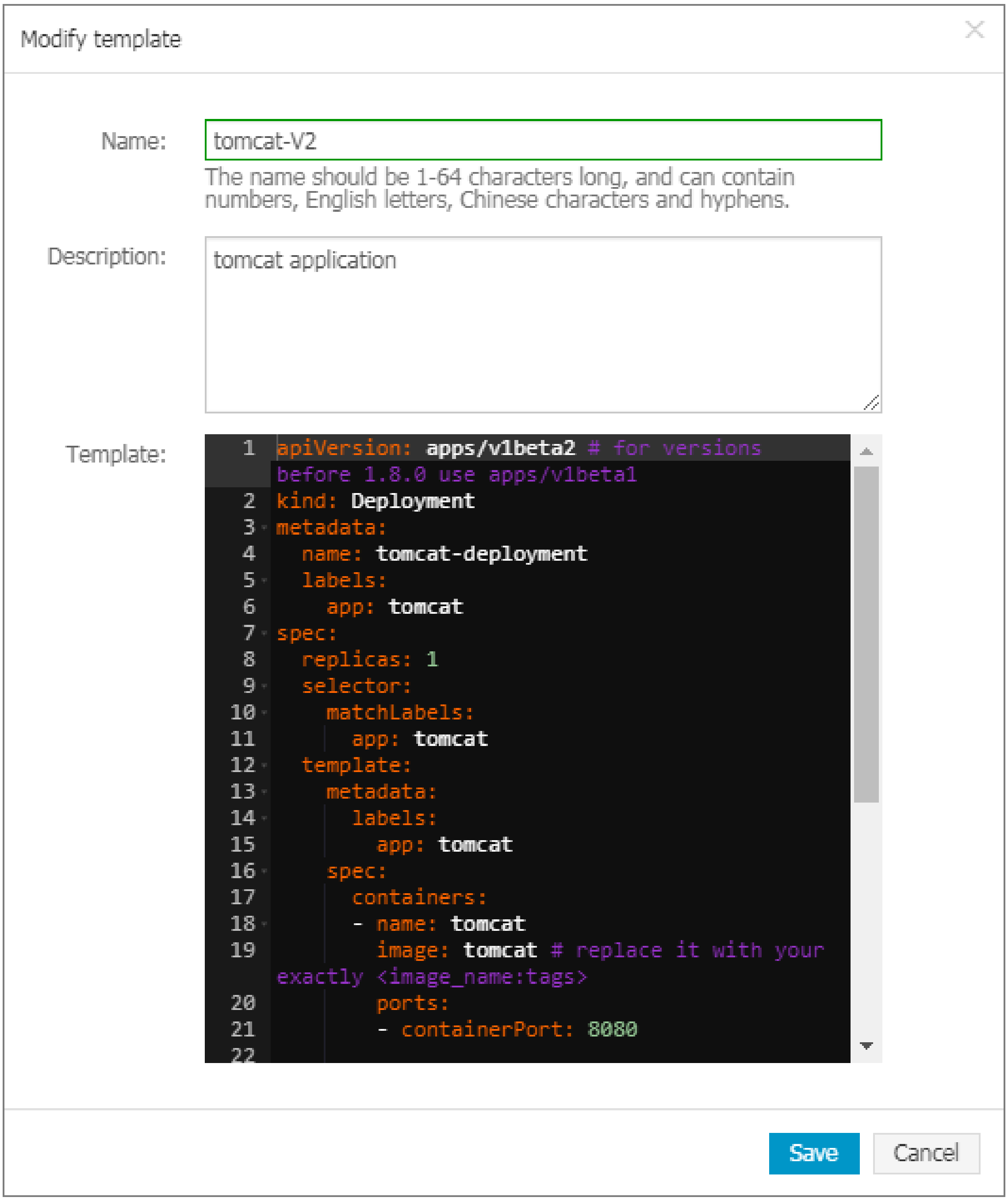
2. Under Kubernetes, click Store > Orchestration Templates. Existing orchestration templates are displayed under My Template.
3. Select a template and click Details.



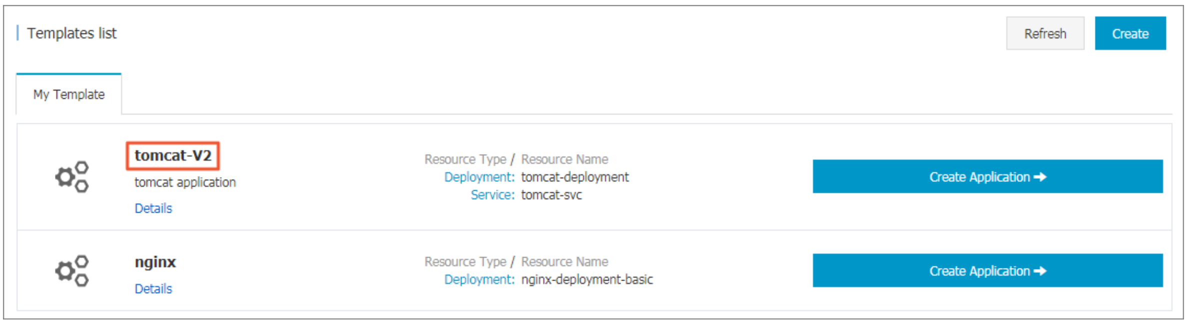
4. Click Edit in the upper-right corner.



5. In the displayed dialog box, edit the name, description, and template, and click Save.



6. Back to the Template List page, under My Template, you can see the template is changed.



1.17.3 Save an existing orchestration template as a new one

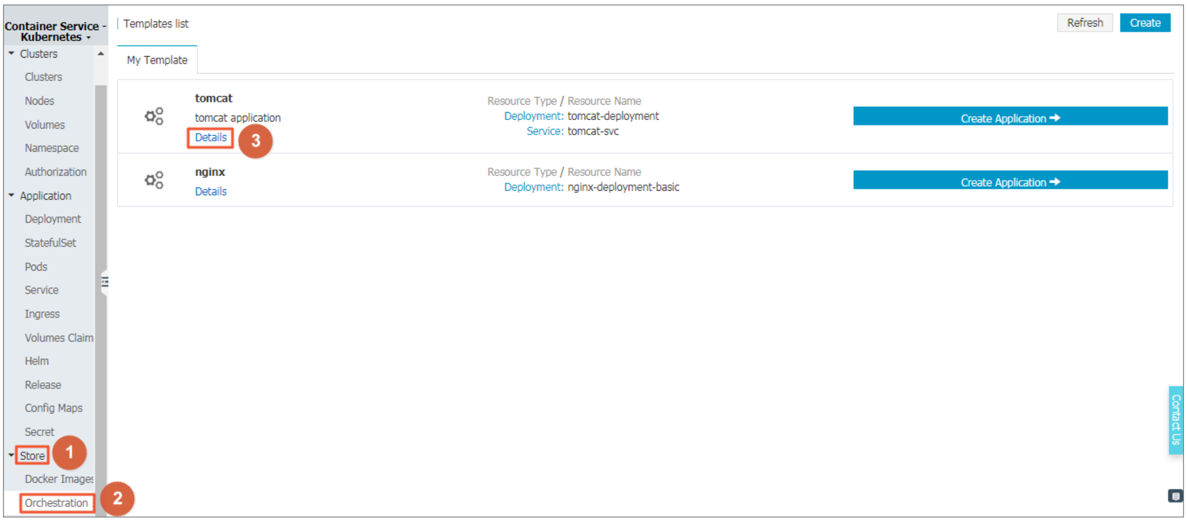
You can save an existing template as a new one.

Prerequisites

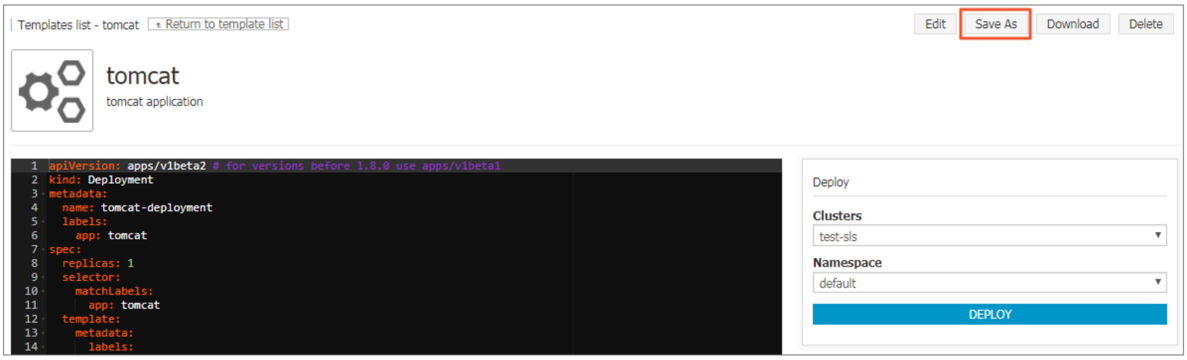
You have created an orchestration template, see *Create an orchestration template*.

Procedure

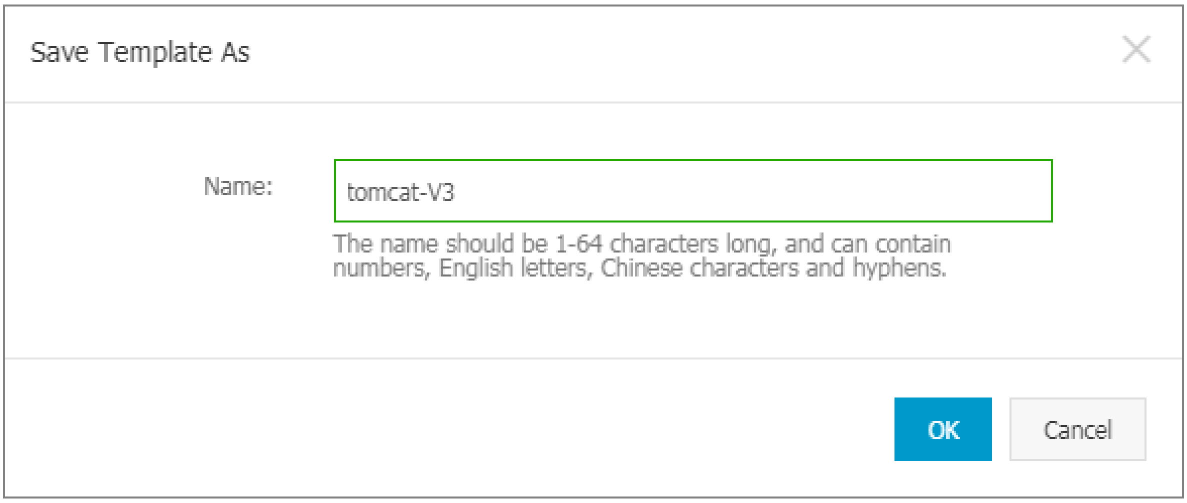
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Store > Orchestration Templates. Existing orchestration templates are displayed under My Template.
3. Select a template and click Details.



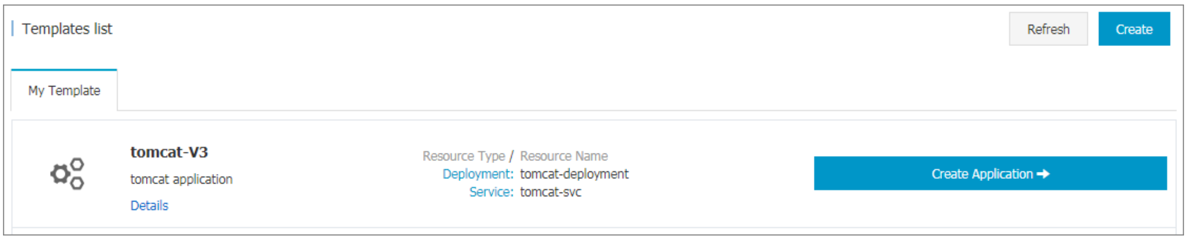
4. You can modify the template and click Save as in the upper-right corner.



5. In the displayed dialog box, configure the template name and click OK.



6. Back to the Template List page, you can see that the saved template is displayed under My Template.



1.17.4 Download an orchestration template

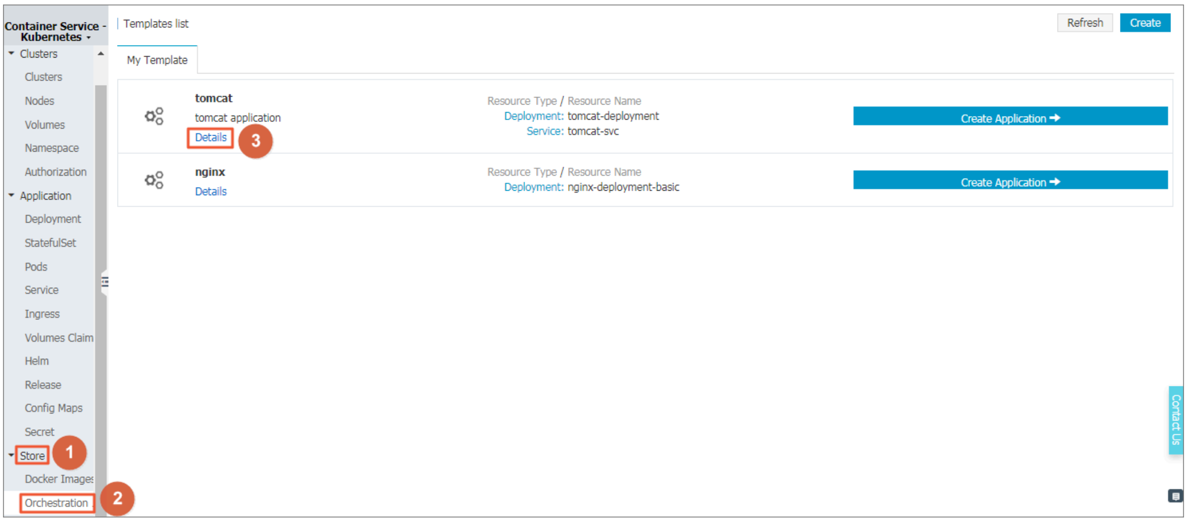
You can download an existing orchestration template.

Prerequisites

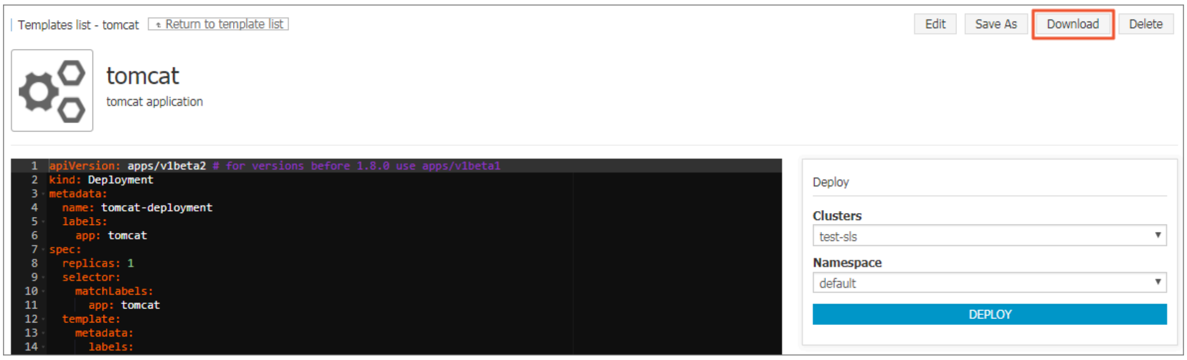
You have created an orchestration template, see *Create an orchestration template*.

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.



1.17.5 Delete an orchestration template

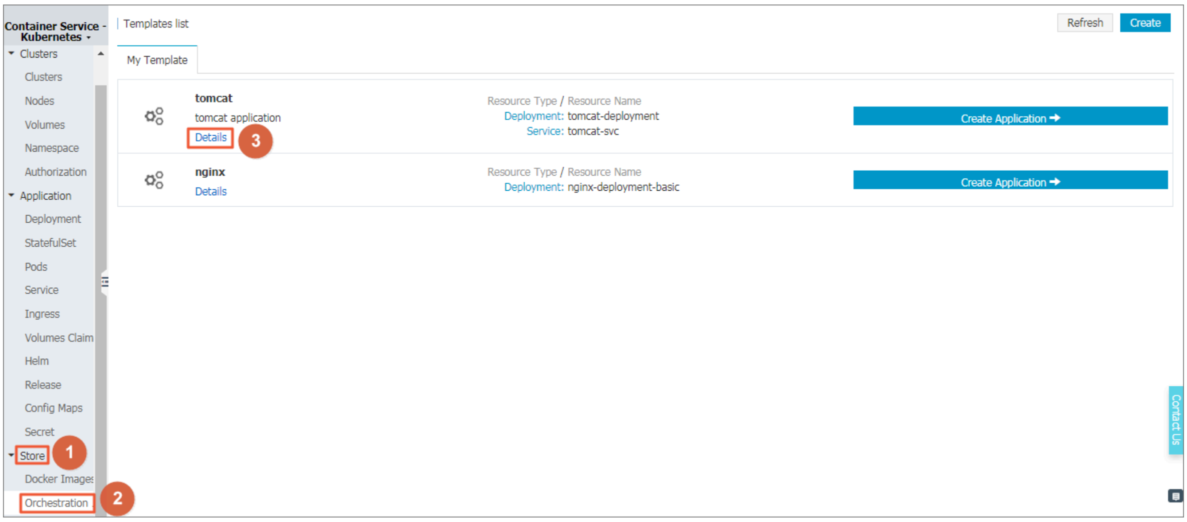
You can delete an orchestration template that is no longer needed.

Prerequisites

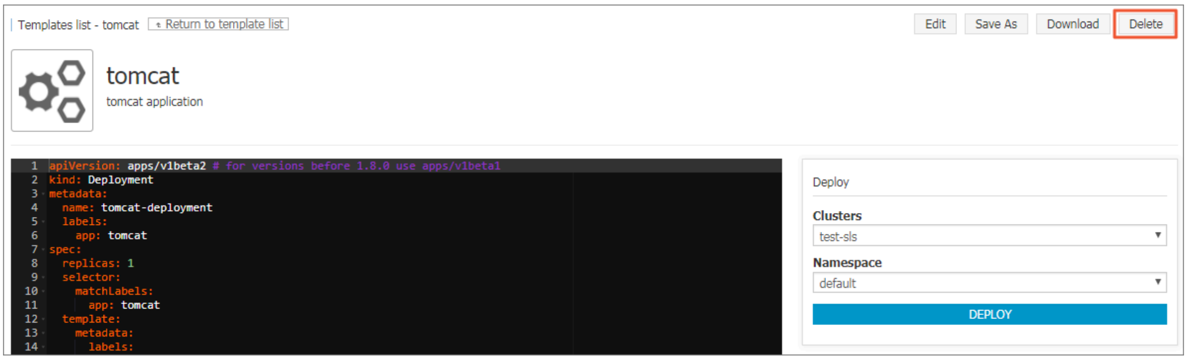
You have created an orchestration template, see *Create an orchestration template*.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click Store > Orchestration Template. Existing orchestration templates are displayed under My Template on the Template list page.
3. Select a template and click Detail.



4. On the detail page of the template, you can click Delete in the upper-right corner.



5. Click Confirm in the displayed dialog box.

1.18 App catalog management

1.18.1 App catalog overview

Microservice is the theme of container era. The application microservice brings great challenge to the deployment and management. By dividing a large single applicatio n into several microservices, the microservice can be independently deployed and extended so as to realize the agile development and fast iteration. Microservice brings great benefits to us. However, developers have to face the management issues of the microservices, such as the resource management, version management, and configuration management. The number of microservices is large because an application is divided into many components that correspond to many microservices.

For the microservice management issues under Kubernetes orchestration, Alibaba Cloud Container Service introduces and integrates with the Helm open-source project to help simplify the deployment and management of Kubernetes applications.

Helm is an open-source subproject in the Kubernetes service orchestration field and a package management tool for Kubernetes applications. Helm supports managing and controlling the published versions in the form of packaging softwares, which simplifies the complexity of deploying and managing Kubernetes applications.

Alibaba Cloud app catalog feature

Alibaba Cloud Container Service app catalog feature integrates with Helm, provides the Helm-related features, and extends the features, such as providing graphic interface and Alibaba Cloud official repository.

The chart list on the App Catalog page includes the following information:

- Chart name: A Helm package corresponding to an application, which contains the image, dependencies, and resource definition required to run an application.
- Version: The version of the chart.
- Repository: The repository used to publish and store charts, such as the official repository stable and incubator.

The information displayed on the details page of each chart may be different and include the following items:

- Chart introduction
- Chart details
- Prerequisites for installing chart to the cluster, such as pre-configuring the persistent storage volumes (pv)
- Chart installation commands
- Chart uninstallation commands
- Chart parameter configurations

Currently, you can deploy and manage the charts in the app catalog by using the Helm tool. For more information, see *Simplify Kubernetes application deployment by using Helm*.

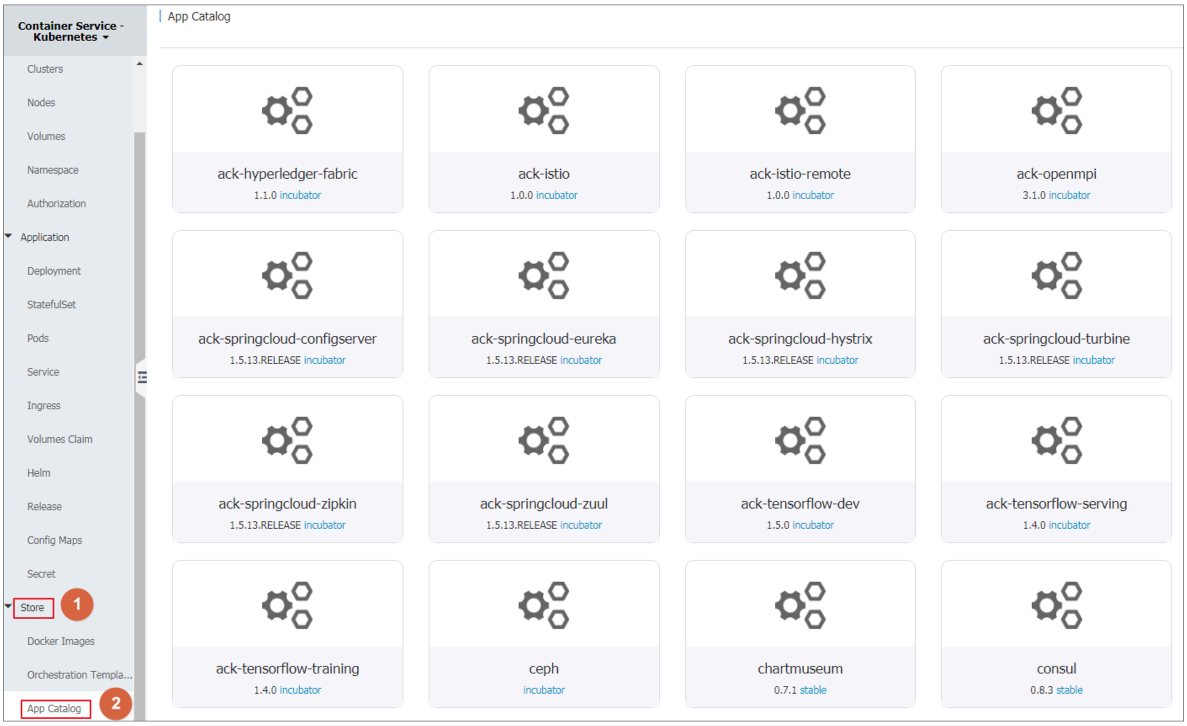
1.18.2 View app catalog list

Procedure

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

2. Under Kubernetes, click Store > App Catalog in the left-side navigation pane.

View the charts on the App Catalog page, each of which corresponds to an application, containing some basic information such as the application name, version, and source repository.



What's next

You can click to enter a chart and get to know the detailed chart information. Deploy the application according to the corresponding information by using the Helm tool. For more information, see *Simplify Kubernetes application deployment by using Helm*.

1.19 Service catalog management

1.19.1 Overview

Applications running on the cloud platform need some basic services such as databases, application servers, and other generic basic softwares. For example, a WordPress application, as a Web application, needs a database service (such as MariaDB) in the backend. Traditionally, you can create the MariaDB service on which the application depends in the WordPress application orchestration, and integrate the MariaDB service with the Web application. To develop applications on the cloud in this way, developers must spend time and energy deploying and configuring the dependent infrastructure softwares, which increases the costs of hosting and migrating applications.

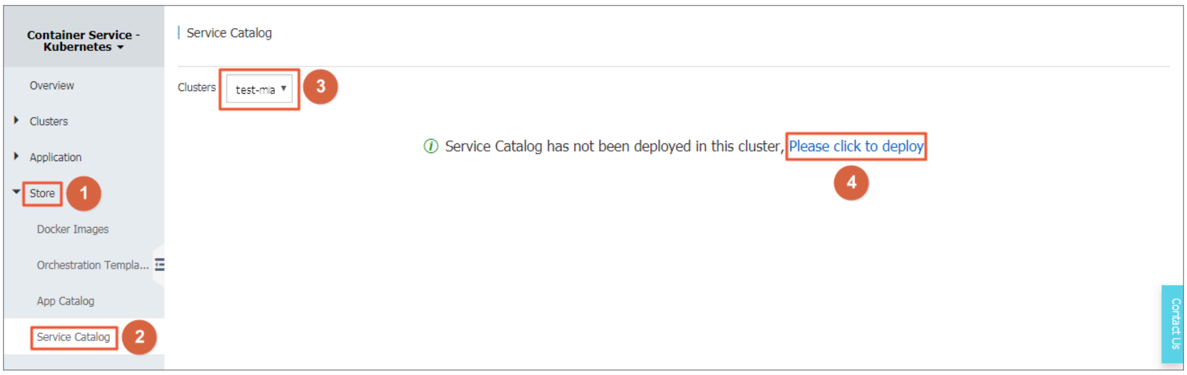
Alibaba Cloud Container Service supports and integrates with the service catalog function. The service catalog function aims to access and manage the service brokers , which allows applications running in Kubernetes clusters to use the managed services offered by service brokers. A series of infrastructure softwares are supported by the service catalog function, which allows the developers to use these softwares as services and focus on the applications, the core of the development, without concerning about the availability and scalability of the softwares or managing the softwares.

The service catalog uses the Open service broker API of Kubernetes to communicate with service brokers, acting as an intermediary for the Kubernetes API server to negotiate the initial provisioning and obtain the credentials necessary for the applications to use the managed services. For more information about the implementation principle of the service catalog, see *Service catalog*.

1.19.2 Enable service catalog function

Procedure

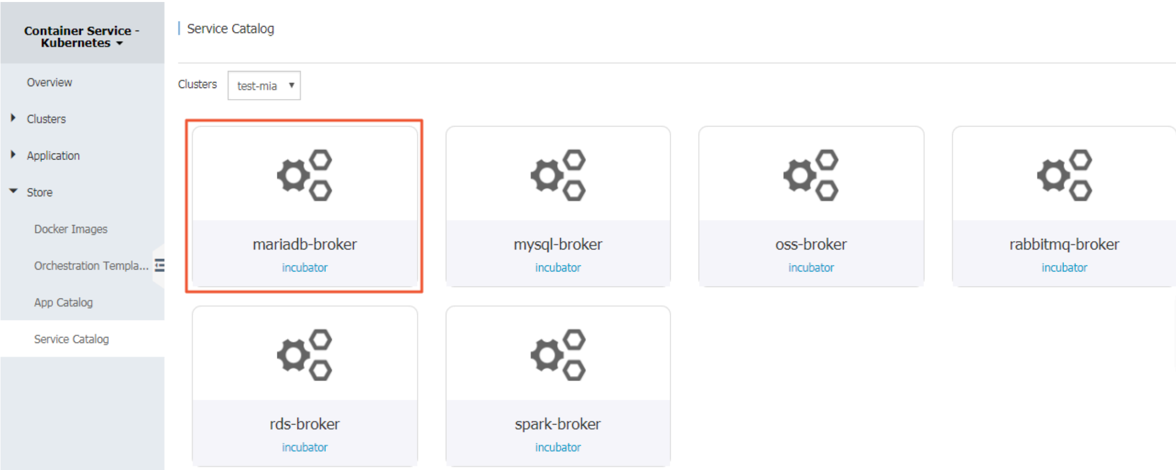
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Store > Service Catalog in the left-side navigation pane. Select the cluster from the Cluster drop-down list in the upper-right corner.
3. If you have not deployed the service catalog, click to install the service catalog as instructed on the page.



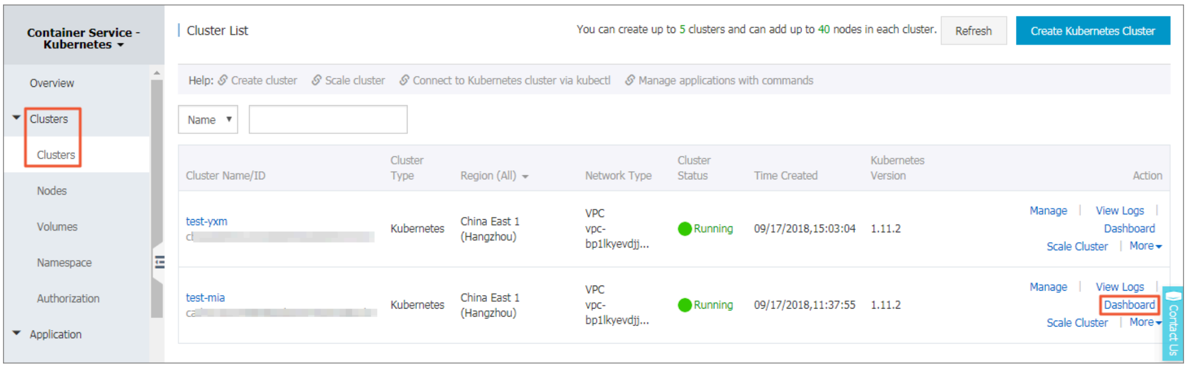
4. After the installation, the service broker, which is installed by default, is displayed on the Service Catalog page. You can click the mariadb-broker to view the details.

Note:

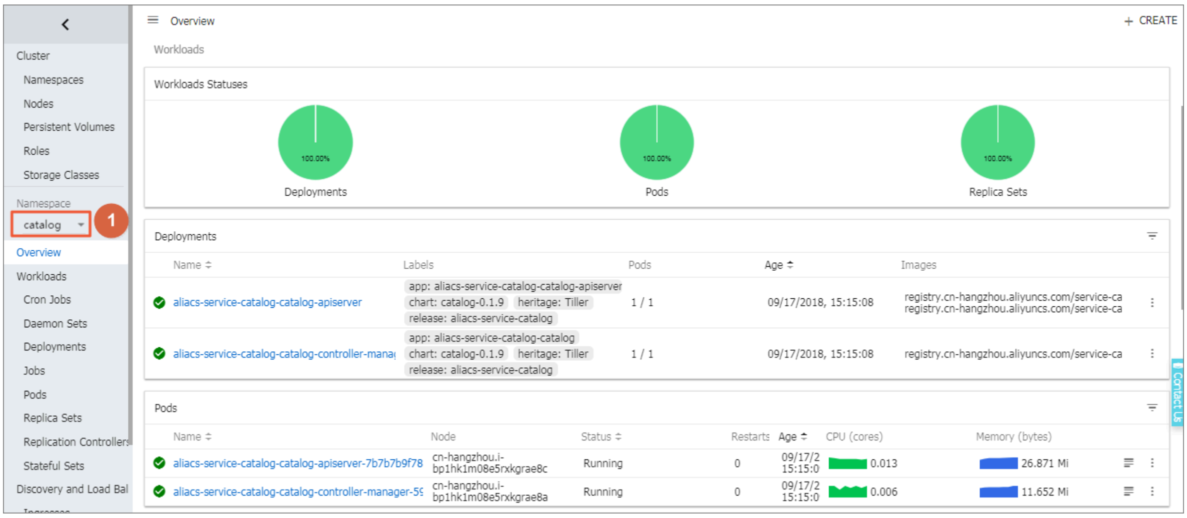
The service catalog is implemented as an extension API server and a controller. After Alibaba Cloud Container Service installs the service catalog function, the namespace catalog is created.



5. Click Clusters in the left-side navigation pane. Click Dashboard at the right of a cluster.



6. In the Kubernetes dashboard, select `catalog` as the Namespace in the left-side navigation pane. You can see the resource objects related to catalog apiserver and controller are installed under this namespace.



What's next

Then, you have successfully enabled the service catalog function. You can create a managed service by using the service broker in the service catalog, and apply the managed service to your applications.