Alibaba Cloud Container Service

User Guide

Issue: 20180917

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 information, supplementary instructions, and other content that the user must understand.	 **Note:** Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental 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]*
{} 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 Overview

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

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

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

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

Limits

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

Related open-source projects

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

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

1.2 Alibaba Cloud Kubernetes vs. self-built Kubernetes

Advantages of Alibaba Cloud Kubernetes

Convenient

- Supports creating Kubernetes clusters 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.

Strong

Function	Description
Network	• High-performance Virtual Private Cloud (VPC) network plug-in. • Supports network policy and flow control. Container Service can provide 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, publishing the business frequently may cause the Ingress to have pressure about configurat ion 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

Function	Description
	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, NAS, and EBS, and provides the standard FlexVolume drive. Self-built Kubernetes clusters cannot use the storage resources on the cloud. Alibaba Cloud Container Service provides the best seamless integration.
O&M	• Integrates with Alibaba Cloud Log Service and CloudMonitor. • Supports auto scaling.
Image repository	• High availability. Supports high concurrency. • Supports speeding up the pull of images. • Supports P2P distribution. The self-built image repository may crash if you pull images from millions of clients at the same time. Enhance the reliability of the image repository by using the Apsara Stack version of Container Service image repository , which reduces the O&M burden and upgrade pressure.
Stable	• Special teams to guarantee the stability of containers. • Each Linux version and Kubernetes version are provided to you after the strict test. Container Service provides the Docker CE to reveal all the details and promotes the repair capabilities of Docker. If you have issues such as Docker Engine hang, network problems, and kernel compatibility, Container Service provides you with the best practices.
High availability	• Supports multiple zones. • Supports backup and disaster recovery.
Technical support	• Provides the Kubernetes upgrade capabiliti es. Supports upgrading a Kubernetes cluster to the latest version with one click.

Function	Description
	• Alibaba Cloud container team is 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 professional personnel.
- The container technology is continuously developing and the version iteration is fast, which requires continuous upgrade and test.

1.3 Clusters

1.3.1 Create a cluster

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

Instructions

During cluster creation, the 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 NAT gateway and Elastic IP (EIP).

- Create a Resource Access Management (RAM) user and the AccessKey. This RAM user has the permissions of querying, creating, and deleting ECS instances, adding and deleting cloud disks, and all the permissions of Server Load Balancer instances, CloudMonitor, VPC , Log Service, and NAS. Kubernetes clusters dynamically create the Server Load Balancer instances, cloud disks, and VPC routing rules according to your configurations.
- Create an intranet Server Load Balancer instance and expose the port 6443.
- Create an Internet Server Load Balancer instance and expose the ports 6443, 8443, and 22. (If you select to enable the SSH logon for Internet when creating the cluster, port 22 is exposed. Otherwise, port 22 is not exposed.)

Prerequisites

Activate the following services: Container Service, Resource Orchestration Service (ROS), and RAM.

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, activate ROS before creating a Kubernetes cluster.

Limits

- The Server Load Balancer instance created with the cluster only supports the Pay-As-You-Go billing method.
- Kubernetes clusters only support the network type VPC.
- By default, each account has a certain quota for the cloud resources they can create. The cluster fails to be created if the quota is exceeded. Make sure you have enough quota before creating the cluster. To increase your quota, open a ticket.
  - By default, each account can create at most five clusters in all regions and add up to 40 worker nodes to each cluster. To create more clusters or nodes, open a ticket. To create more clusters or nodes, open a ticket.
  - By default, each account can create at most 100 security groups.
  - By default, each account can create at most 60 Pay-As-You-Go Server Load Balancer instances.

— By default, each account can create at most 20 EIPs.

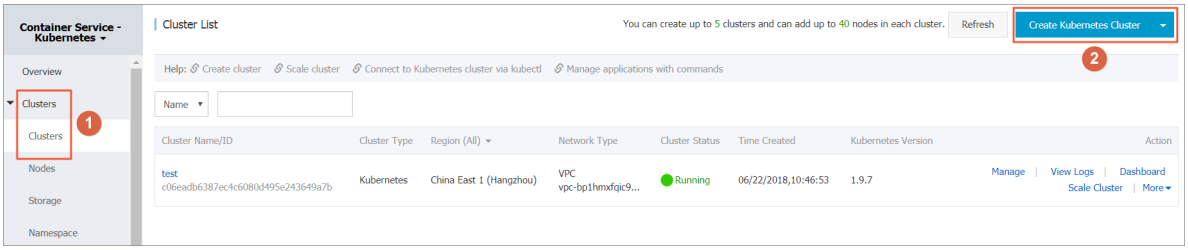
- Limits for ECS instances are as follows:

  — Only support the CentOS operating system.

  — Creating Pay-As-You-Go and Subscription ECS instances is supported.

Procedure

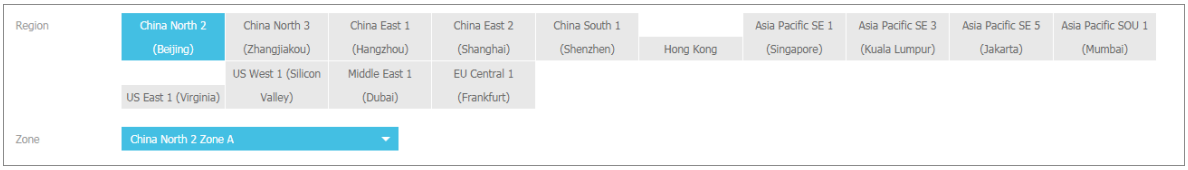
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane to enter the Cluster List page.
3. Click **Create Kubernetes Cluster** in the upper-right corner.



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 in which the cluster resides.



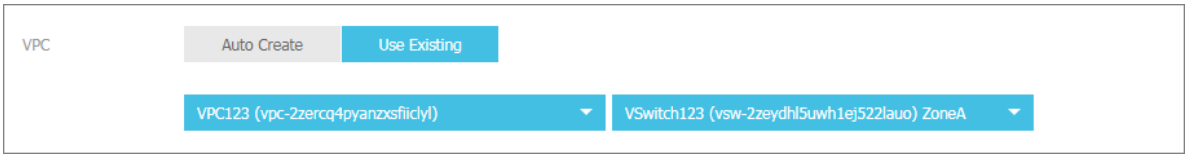
6. Set the cluster network type. Kubernetes clusters only support the VPC network type.

   You can select **Auto Create** to create a **Virtual Private Cloud (VPC)** together with the Kubernetes cluster or **Use existing** to use an existing VPC. With **Use Existing** selected, choose the VPC and VSwitch from the appeared drop-down list.

   - With **Auto Create** selected, the system automatically creates a NAT gateway for your VPC when the cluster is created.
   - With **Use Existing** selected, if the selected VPC already has a NAT gateway, Container Service uses the existing 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, clear the **Configure SNAT for VPC** check box.

Note:

If you select to not automatically create a NAT gateway, configure the NAT gateway on your own to implement the VPC public network environment with secure access, or manually configure the SNAT. Otherwise, instances in the VPC cannot access public network normally, which leads to cluster creation failure.

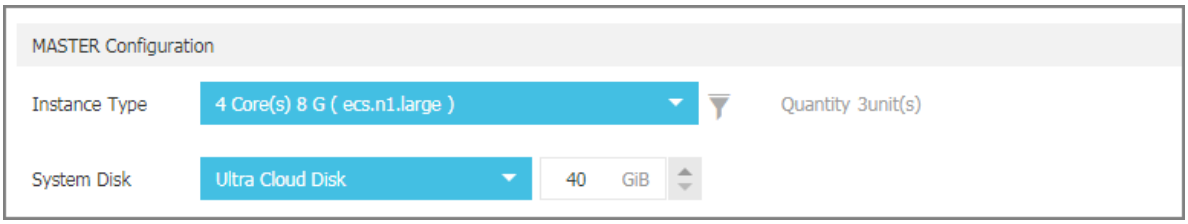


7. Configure the node type, Pay-As-You-Go and Subscription types are supported.
8. Configure the master nodes.

   Select the generation, family, and type for the master nodes.

Note:

- Currently, master nodes only support CentOS operating system.
- Currently, you can only create three master nodes.
- Supports mounting system disks for the master node, SSD and high-efficiency cloud disks are supported.

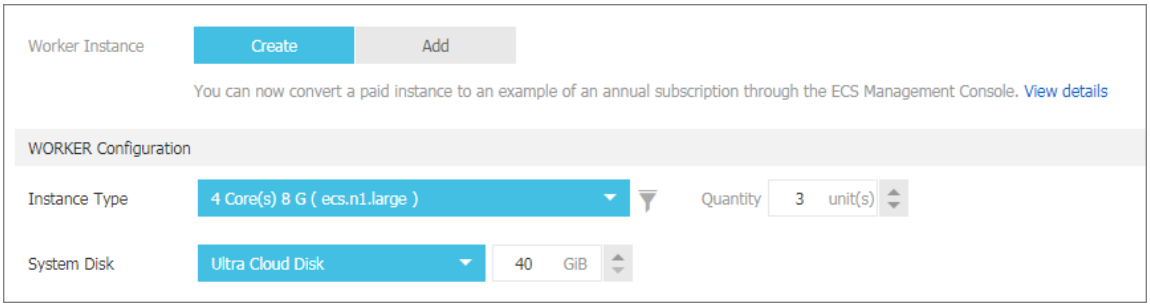


9. Configure the worker nodes. Select whether to create a worker node or add an existing ECS instance as the worker node.

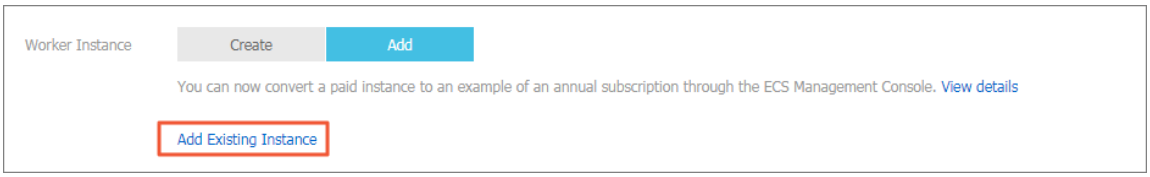
Note:

- Currently, worker nodes only support the CentOS operating system.
- Each cluster can contain up to 37 worker nodes. To create more nodes, open a ticket.
- Supports mounting system disks for the worker node, SSD, high-efficiency, and basic cloud disks are supported.

a. If you want to add an instance, you must generation, family, and type for the worker node., and number for the worker nodes (in this example, select to create one worker node).



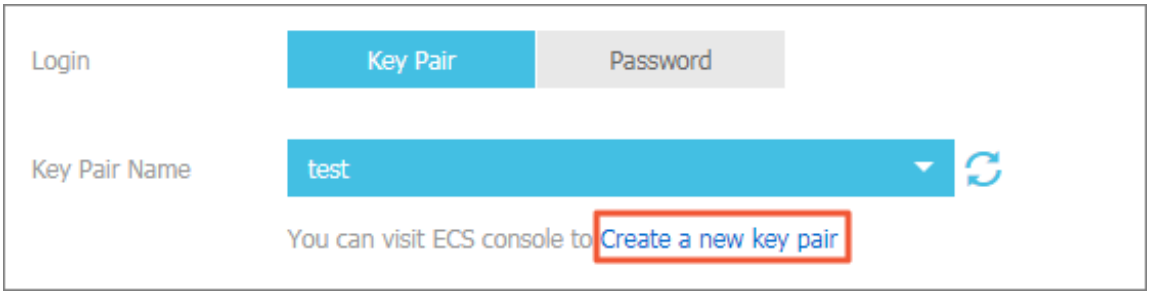
b. To add an existing ECS instance as the worker node, you must create an ECS instance in the current region in advance.



10. Configure the logon mode.

- Set the secret.

  Select the key pair logon mode when creating the cluster, click **New Key Pair**. Go to the ECS console, and create a key pair, 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**: Configure the node logon password.

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

11. Configure the **Pod Network CIDR** and **Service CIDR**.

> **Note:**
> This option is available when you select to **use an existing** VPC.

Specify the **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, and you cannot modify the values after the cluster is created. Service address segment cannot be repeated with the Pod address segment. Besides, the service CIDR block cannot overlap with the pod CIDR block. For more information about how to plan the Kubernetes CIDR blocks, see *Plan Kubernetes CIDR blocks under VPC*.

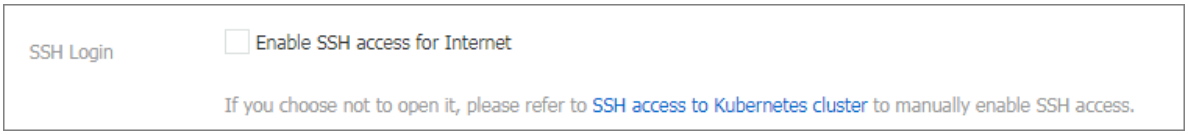
12. Set whether to configure a SNAT gateway for a private network.

Note:

SNAT must be configured if you select **Auto Create** VPC. If you select **Use existing** VPC, you can select whether to automatically configure SNAT gateway. If you select not to configure SNAT automatically, you can configure the NAT gateway to implement VPC security access to the public network. You can also configure SNAT manually. Otherwise, the VPC cannot access the public network.

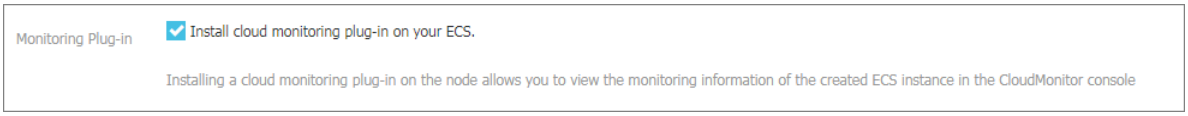
13. Select whether to enable SSH logon for Internet.

- With this check box selected, you can access the cluster by using SSH.
- If this check box is not selected, you cannot access the cluster by using SSH or connect to the cluster by using kubectl. To access the cluster by using SSH, manually bind EIP to the ECS instance, configure security group rules, and open the SSH port (22). For more information, see *Access Kubernetes clusters by using SSH*.



14. Sets whether the cloud monitoring plug-in is enabled.

You can select to install the cloud monitoring plug-in on the ECS instance and then view the monitoring information of the created ECS instance in the CloudMonitor console.



15. Select to add the IP addresses of the ECS instances to the RDS instance whitelist.

It facilitates the ECS instances to access the RDS instances.

Note:

This option is available if you are **using an existing** VPC. The ECS instance must be in the same region and same VPC environment as the RDS instance so that the IP address of the ECS instance can be added to the RDS instance whitelist.

a. Click **Select RDS Instances**.

b. The Add to RDS instance whitelist dialog box appears. Select the RDS instances and then click **OK**.

16. Select whether to enable the advanced configurations.

a. Enable the network plug-ins, Flannel and Terway network plug-ins are supported.

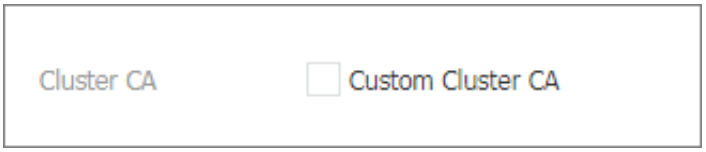
- Flannel: The Flannel cni plug-in for simple and stable communities.
- Terway: Alibaba Cloud Container Service self-developed network plug-in, which supports Alibaba Cloud flexible network card to be distributed to the container, and supports Kubernetes `NetworkPolicy` to define the inter-container access policy. Supports bandwidth limiting for the separate containers. Currently it is in the public beta.

b. Set the number of nodes pod, which is the maximum number of pods that can be run by a single node. We recommend to maintain the default value.

c. Select whether or not to use the **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 to deploy the environment you need quickly. For example, the image deployed and tested based on the CentOS 7.2 LAMP .

d. Select whether to use **custom cluster CA**. With this check box selected, the CA certificate can be added to the Kubernetes cluster, which enhances the security of information exchange between server and client.



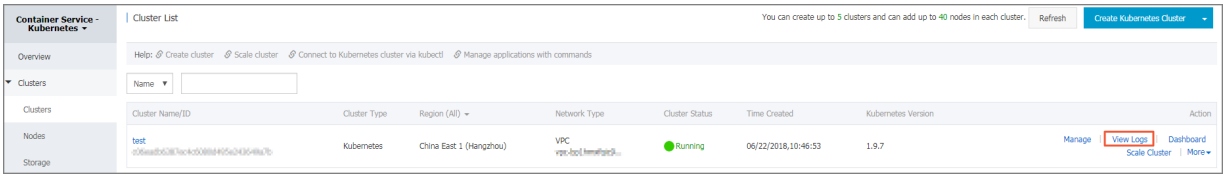
17. Click **Create cluster** to start the deployment.

Note:

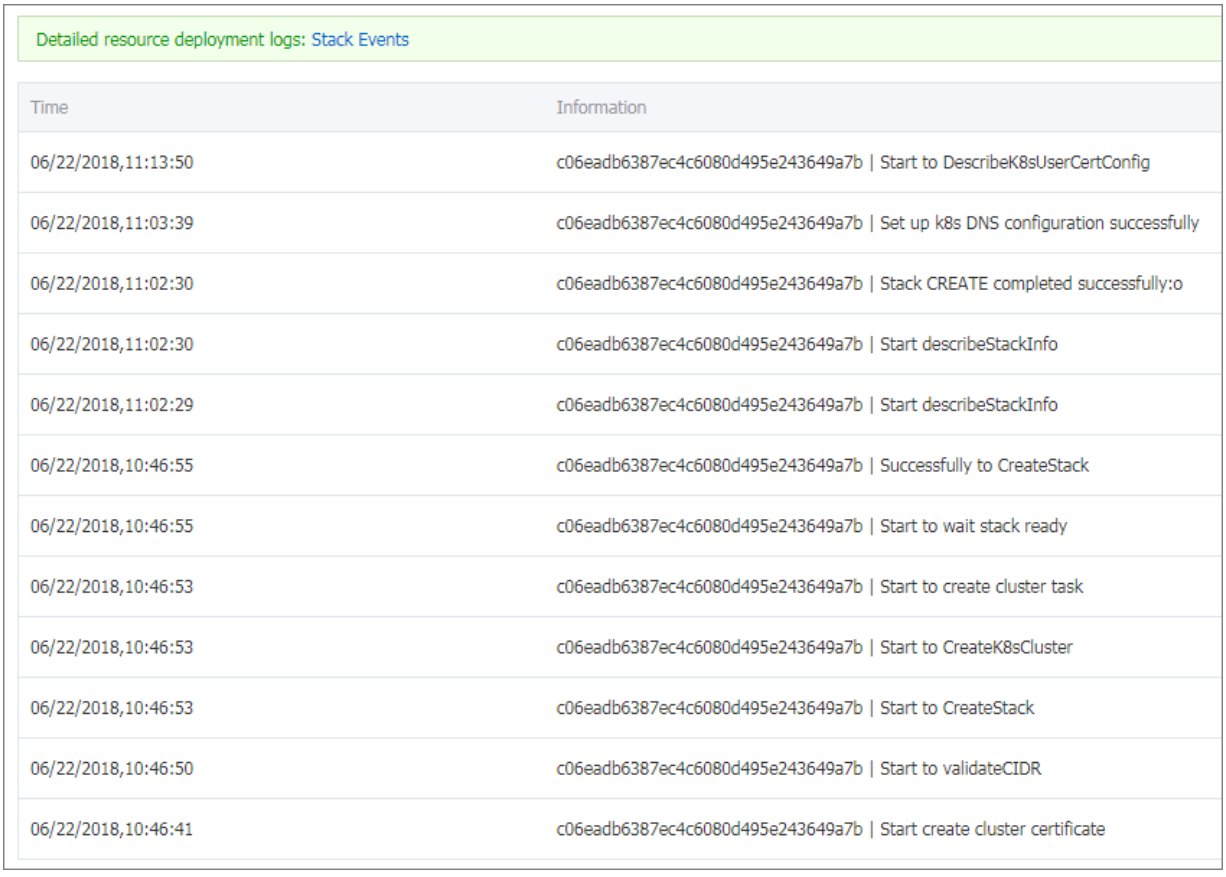
Creating a Kubernetes cluster with multiple nodes lasts more than 10 minutes.

Subsequent operations

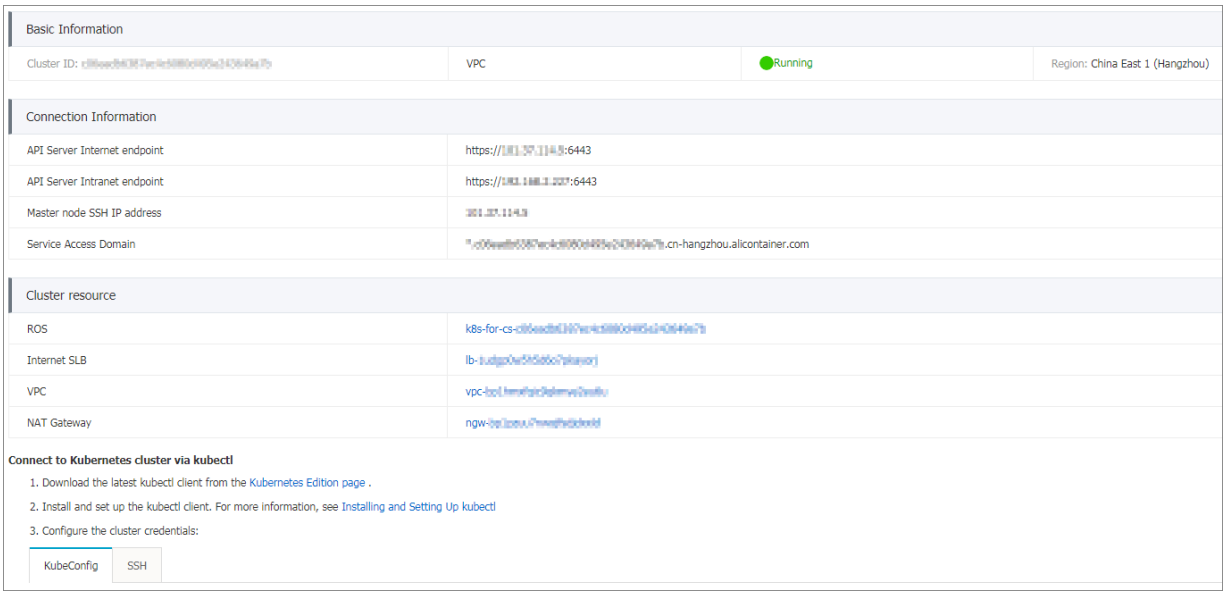
After the cluster is successfully created, you can view the cluster in the Kubernetes Cluster List of the Container Service console.



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



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



In the Connection Information section:

- **API Server Internet endpoint**: The address and port used by the Kubernetes API server to provide the service for the Internet. You can use kubectl or other tools on the user terminal by means of this service to manage the cluster.
- **API Server Intranet endpoint**: The address and port used by the Kubernetes API server to provide the service for the intranet. This IP address is the address of the Server Load Balancer instance, and three master nodes in the backend are providing the service.
- **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 service in the cluster with access domain name for testing. The suffix of the service access domain name is `<cluster_id>.<region_id>.alicontainer.com`.

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

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

Welcome to Alibaba Cloud Elastic Compute Service !

[root@iZbp1d7yvpa3j183u0urllZ ~]# 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@iZbp1d7yvpa3j183u0urllZ ~]#
```

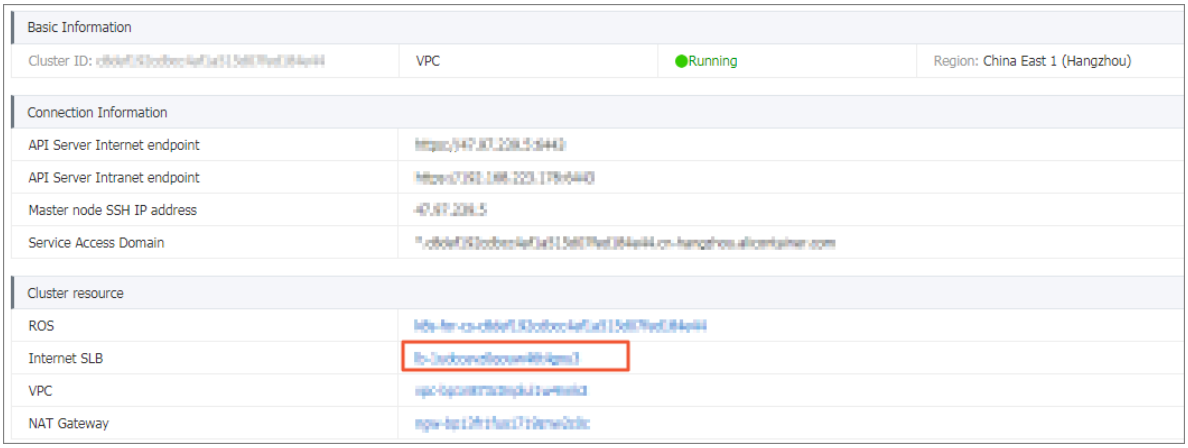
As shown in the preceding figure, the cluster has four nodes, including three master nodes and one worker node configured when creating the cluster.

1.3.2 Access Kubernetes clusters by using SSH

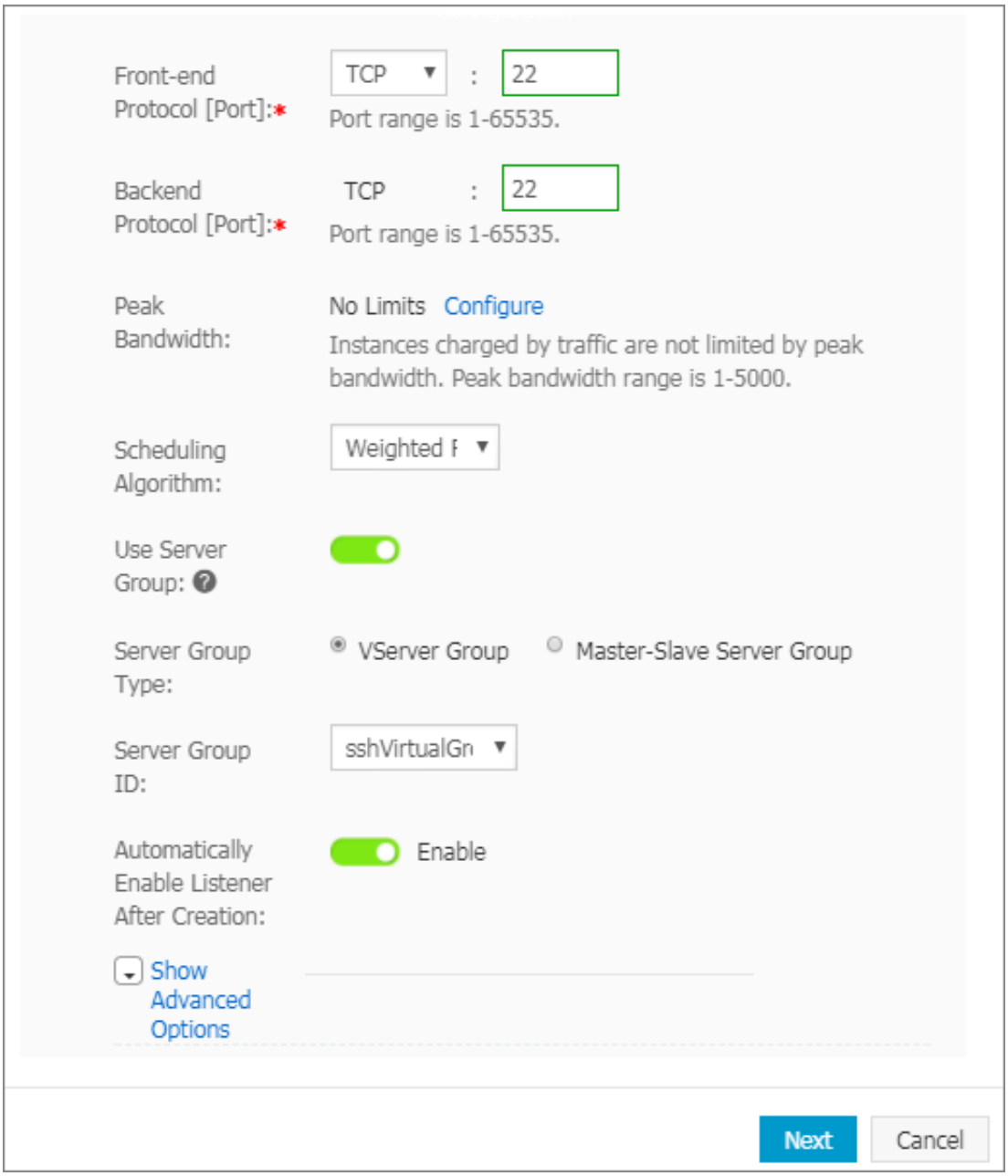
If you select not to enable SSH access for Internet when creating the Kubernetes cluster, you cannot access the Kubernetes cluster by using SSH or connect to the Kubernetes cluster by using kubectl. To access the cluster by using SSH after creating the cluster, manually bind Elastic IP (EIP) to the Elastic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

Procedure

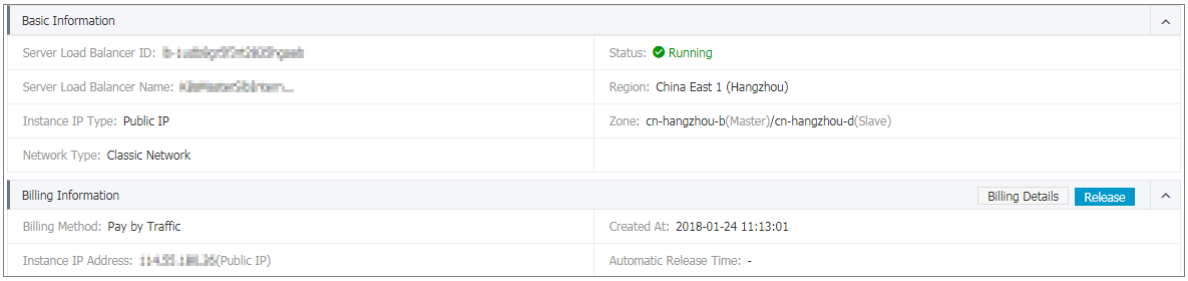
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane.
3. Click **Manage** at the right of the cluster.
4. In **Cluster resource**, click the ID of the Internet SLB. Then, you are redirected to the Instance Details page of your Internet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



5. Click **Listeners** in the left-side navigation pane and then click **Add Listener** in the upper-right corner.
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.



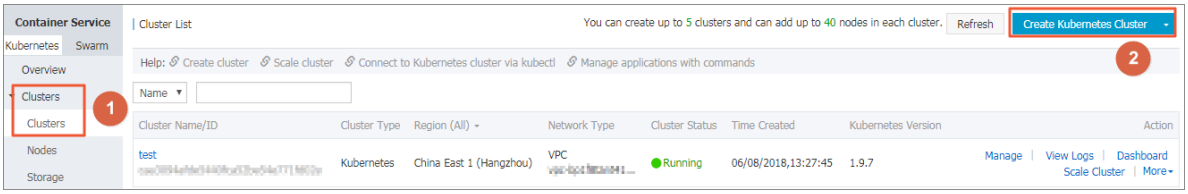
1.3.3 Access Kubernetes clusters by using SSH key pairs

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

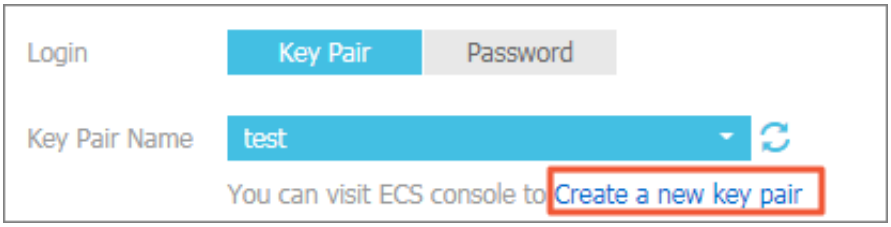
Context

Procedure

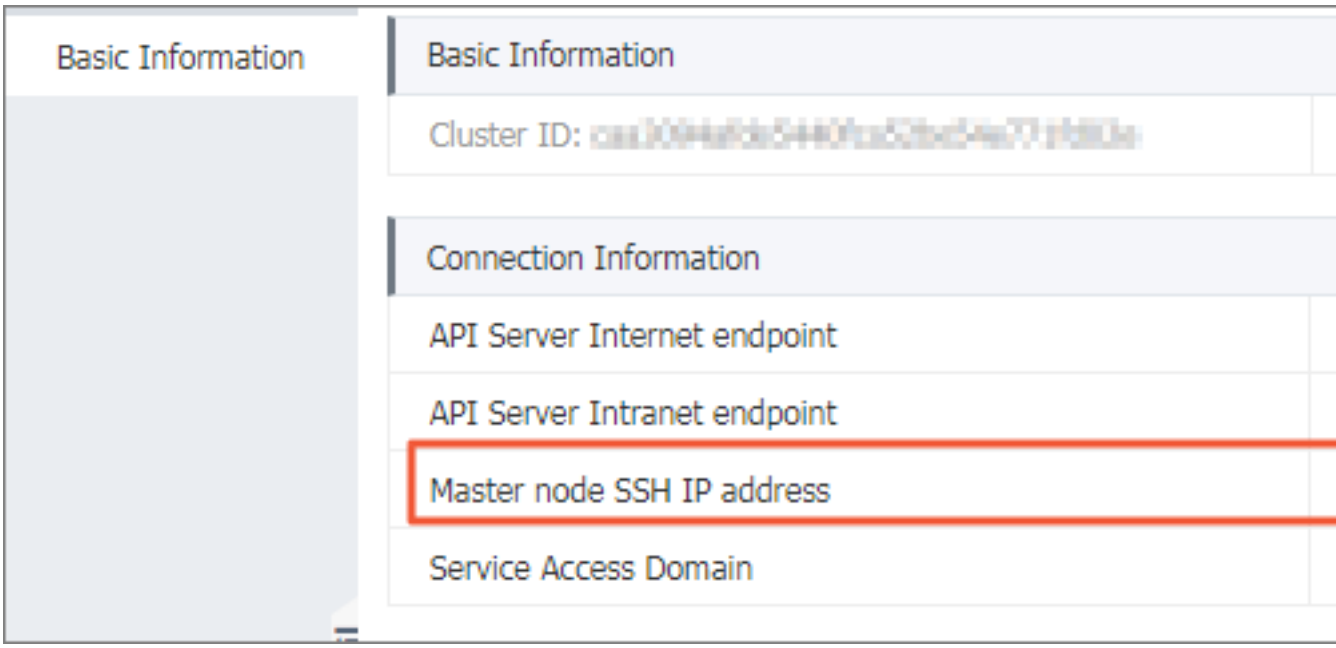
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Clusters** in the left-side navigation pane.
3. Click **Create Kubernetes Cluster** in the upper-right corner.



4. Select Key Pair in the Login field. Complete the other configurations. For more information, see *Create a Kubernetes cluster*. Then, click **Create**.
   1. If you have created key pairs in the Elastic Compute Service (ECS) console, select a key pair from the Key Pair Name drop-down list.
   2. If you have no key pair, click **Create a new key pair** to create one in the ECS console. For more information, see *Create an SSH key pair*.



5. After the cluster is created, click **Manage** at the right of the cluster on the Cluster List page. View the **Master node SSH IP address** under Connection Information.



6. Download the *.pem* private key file. Complete the configurations based on your local operating system environment, such as Windows or Linux. For more information, see *#unique_14*. Take Linux as an example.
   a) Find the path where your downloaded .pem private key file is stored on your local machine. For example, `/root/xxx.pem`.
   b) Run the following command to modify the attributes of the private key file: `chmod 400 [ path where the .pem private key file is stored on the local machine ]`. For example, `chmod 400 /root/xxx.pem`.
   c) Run the following command to connect to the cluster: `` `ssh -i [path where the .pem private key file is stored on the local machine] root@[master-public-ip]`. Wherein, master-public-ip is the master node SSH IP address. For example, `ssh -i /root/xxx.pem root@10.10.10.100`.

1.3.4 Connect to a Kubernetes cluster by using kubectl

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

Procedure

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

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

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

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

```
mkdir $HOME/.kube
scp root@<master-public-ip>:/etc/kubernetes/kube.conf $HOME/.kube/config
```

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

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

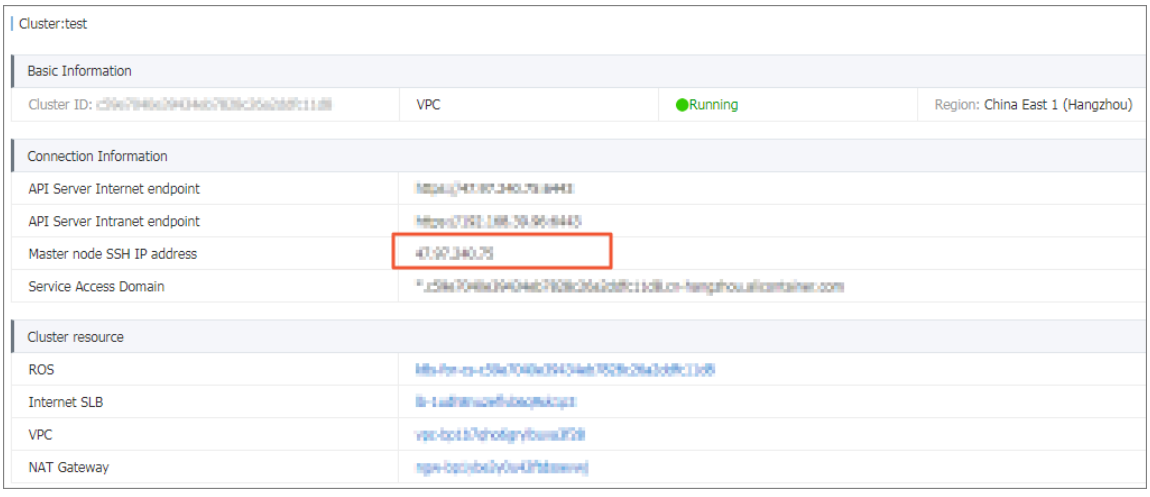
You can check the cluster `master-public-ip` on the cluster information page.

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

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

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

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



1.3.5 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

Instructions

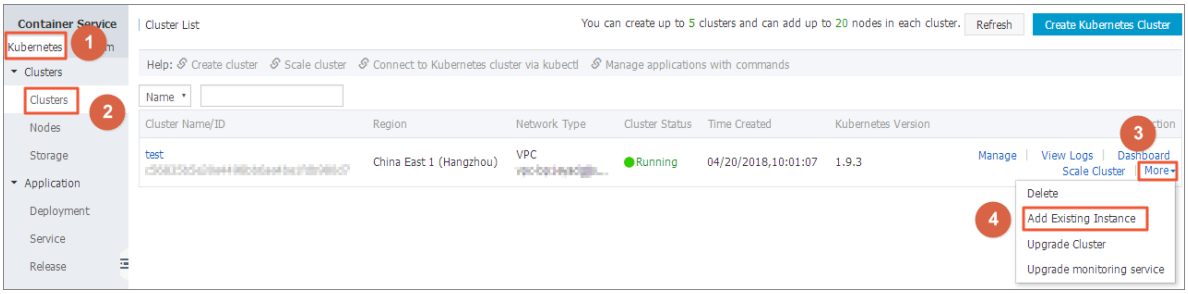
- By default, each cluster can contain up to 40 nodes. To add more nodes, open a ticket.
- The ECS instance to be added must be in the same Virtual Private Cloud (VPC) region as the cluster.
- When adding an existing instance, make sure that your instance has an Elastic IP (EIP) for the VPC network type,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

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**.

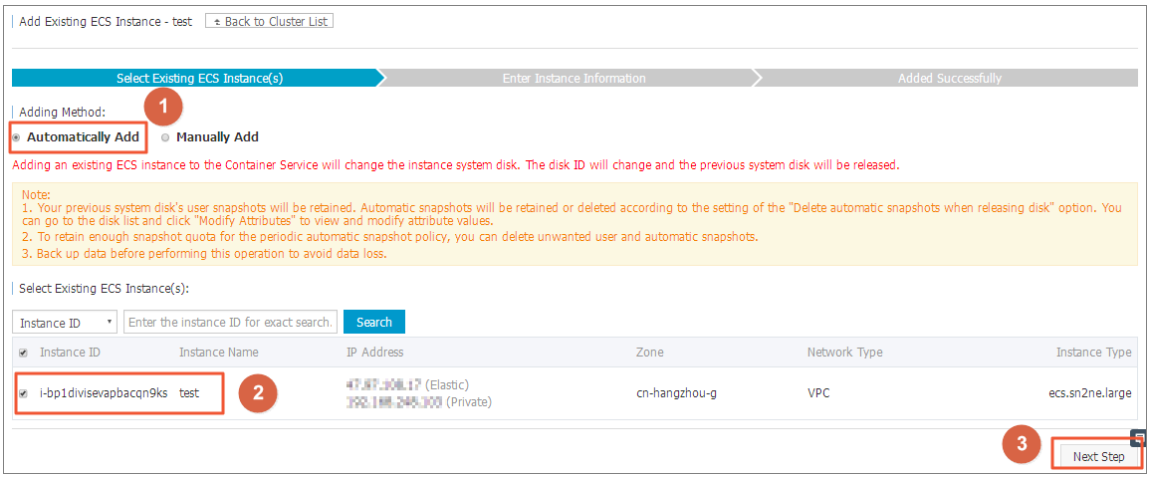
   On the Add Existing ECS Instance page and you can automatically or manually add an existing instance.

   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.



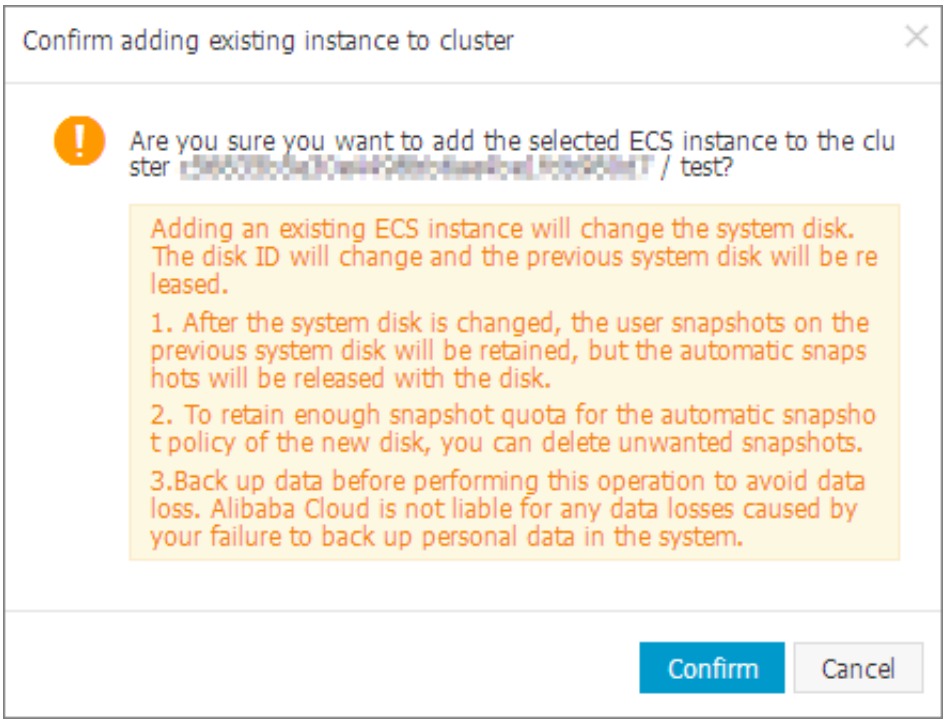
4. Select Automatically Add to add multiple ECS instances at a time.

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



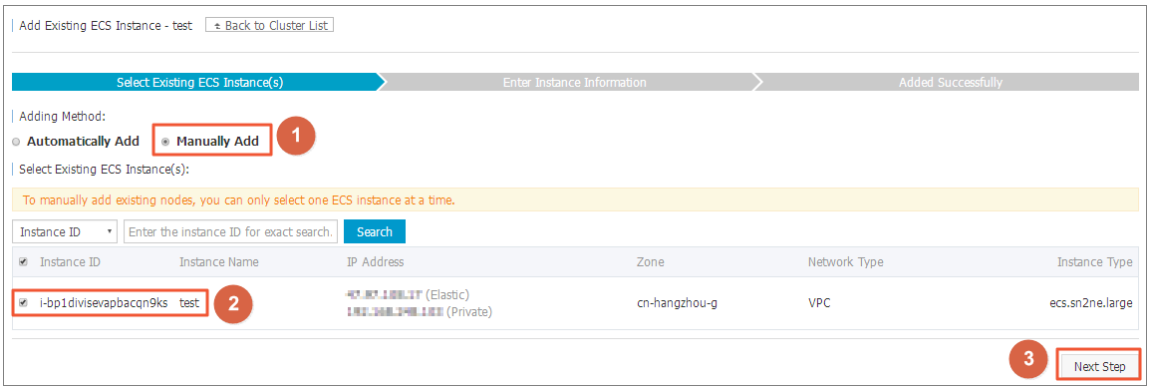
b) Enter the instance information, set the logon password, and then click **Next Step**.

c) In the displayed dialog box, click **OK**, the selected ECS instance is automatically added to the cluster.

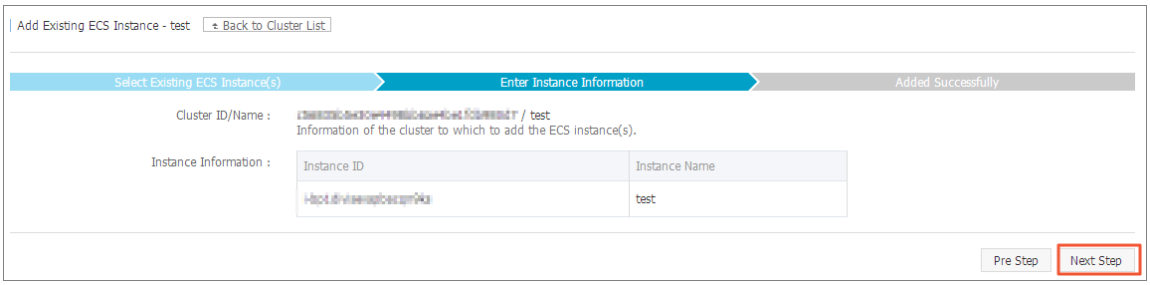


5. You can also select Manually Add to manually add an existing ECS instance to the cluster.

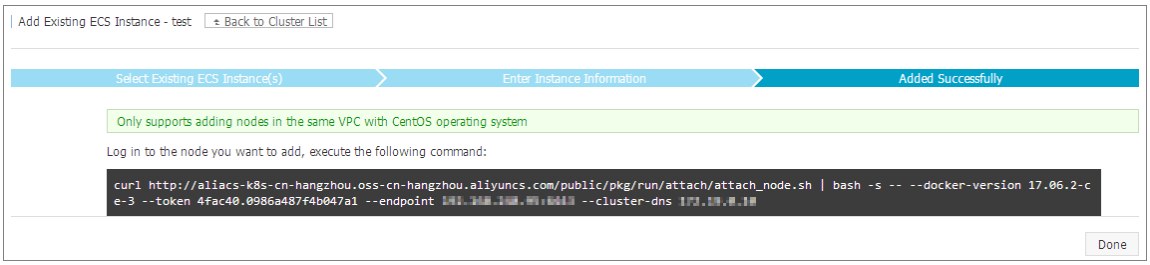
a) Select the ECS instance to be added and then click **Next Step**. You can only add one ECS instance at a time.



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

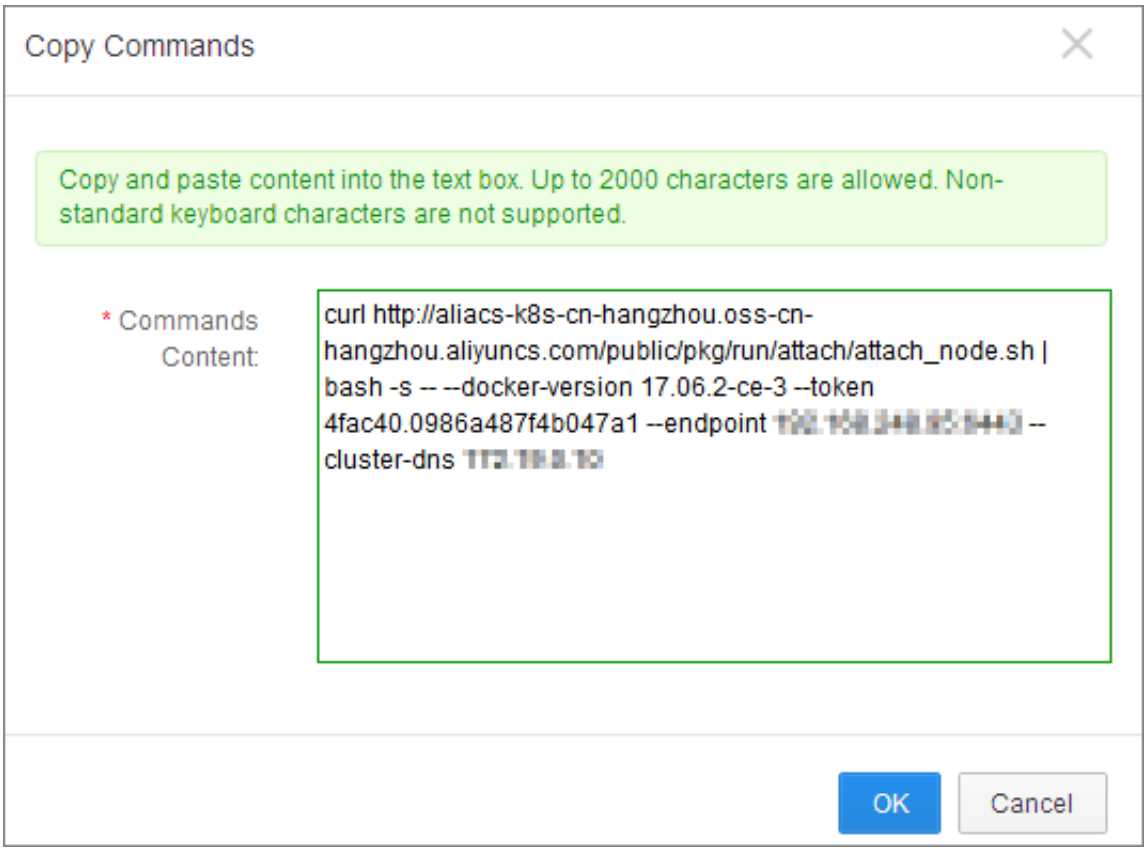


c) Go to the Add Existing ECS Instance page and copy the command.



d) Log on to the *ECS console*. Select the region in which the cluster resides.

e) 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.



f) 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.

1.3.6 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.

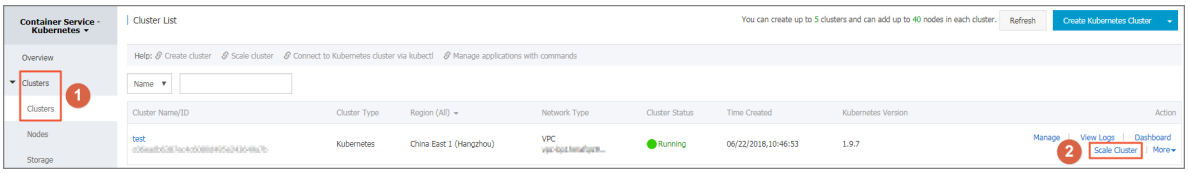
Context

Instructions

- Currently, Container Service only supports manually scaling in and out a cluster and does not support auto scaling.
- Currently, Container Service does not support scaling in and out the master nodes in a cluster.
- Container Service only supports scaling in the worker nodes that are created when you create the cluster or added after you scale out the cluster. The worker nodes that are added as existing *Add an existing ECS instance* when you create the cluster 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.

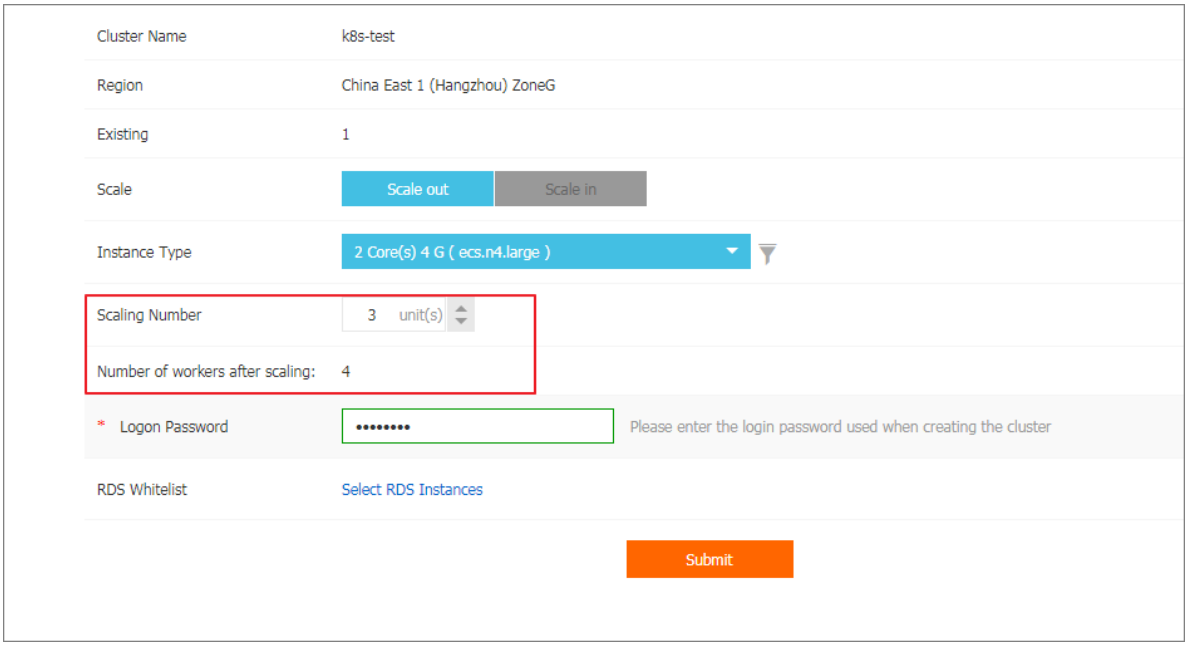
Procedure

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

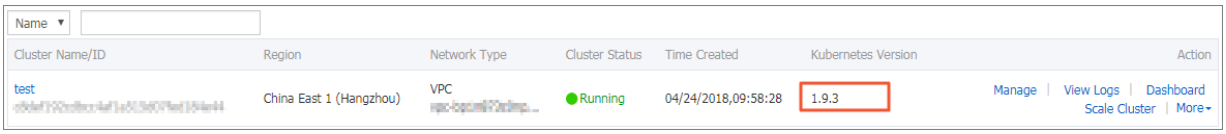
What's next

After scaling is complete, go to the Kubernetes Clusters Node List page to view that the number of worker nodes changes from one to four.

1.3.7 Upgrade a cluster

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

View the Kubernetes version of your cluster in the Kubernetes cluster list.



Instructions

- To upgrade the cluster, make sure your machine can access the Internet to download the necessary software packages.
- The upgrade may fail. We recommend that you back up snapshots before upgrading the cluster to guarantee your data security. For how to create a snapshot, see *Create snapshots*.
- During the upgrade, your applications are not affected, but we recommend that you do not manage the cluster by using kubectl or the Container Service console. The upgrade lasts 5–15 minutes. The cluster status changes to Running after the upgrade.

Prerequisites

Check the health status of the cluster before upgrading the cluster. Make sure the cluster is healthy.

Log on to the master node. For more information, see *Access Kubernetes clusters by using SSH* and *Connect to a Kubernetes cluster by using kubectl*.

1. Run the command `kubectl get cs`. Make sure all the 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"}
```

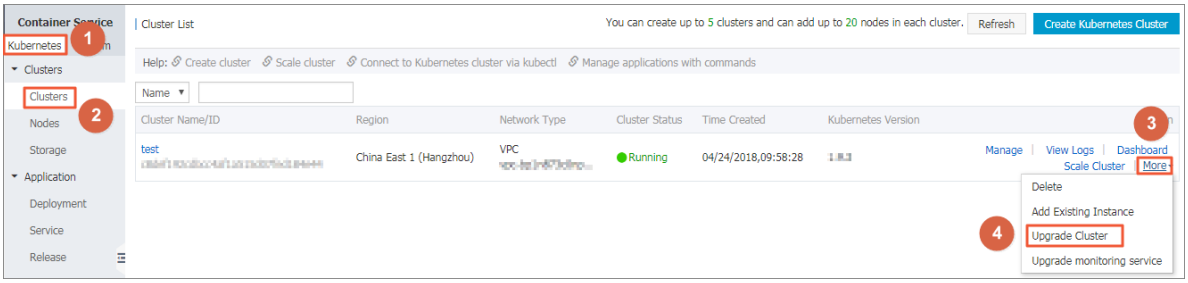
2. Run the command `kubectl get nodes`. Make sure all the nodes are in the Ready status.

```
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 nodes are abnormal, you can fix them by yourself or open a ticket to ask Alibaba Cloud engineers to fix them.

Procedure

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



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

   The system starts to upgrade the Kubernetes version.

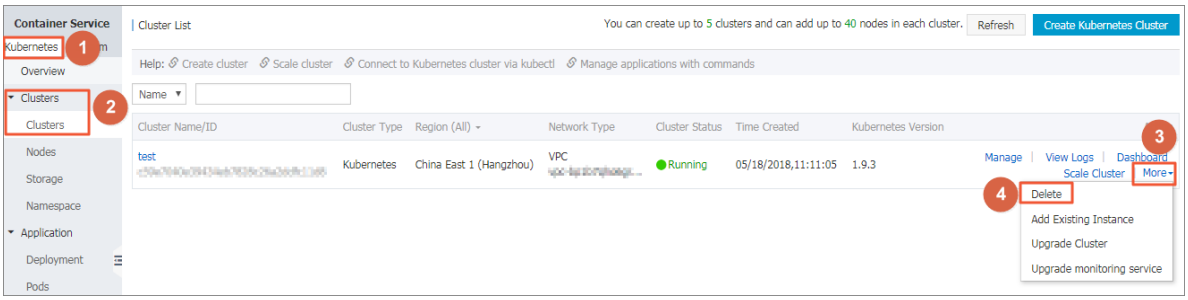
After the upgrade, you can check the Kubernetes version of this cluster in the Kubernetes cluster list to make sure whether or not the upgrade is successful.

1.3.8 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. Click Kubernetes > **Clusters** in the left-side navigation pane.
3. Click**More at the right of the cluster and then select** > **Delete**.



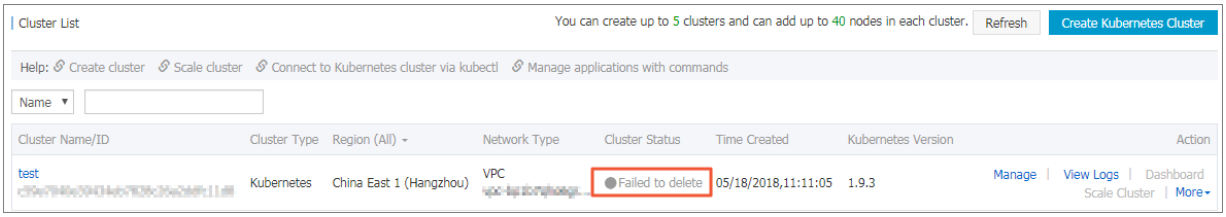
What's next

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 these 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 manually created resources manually.

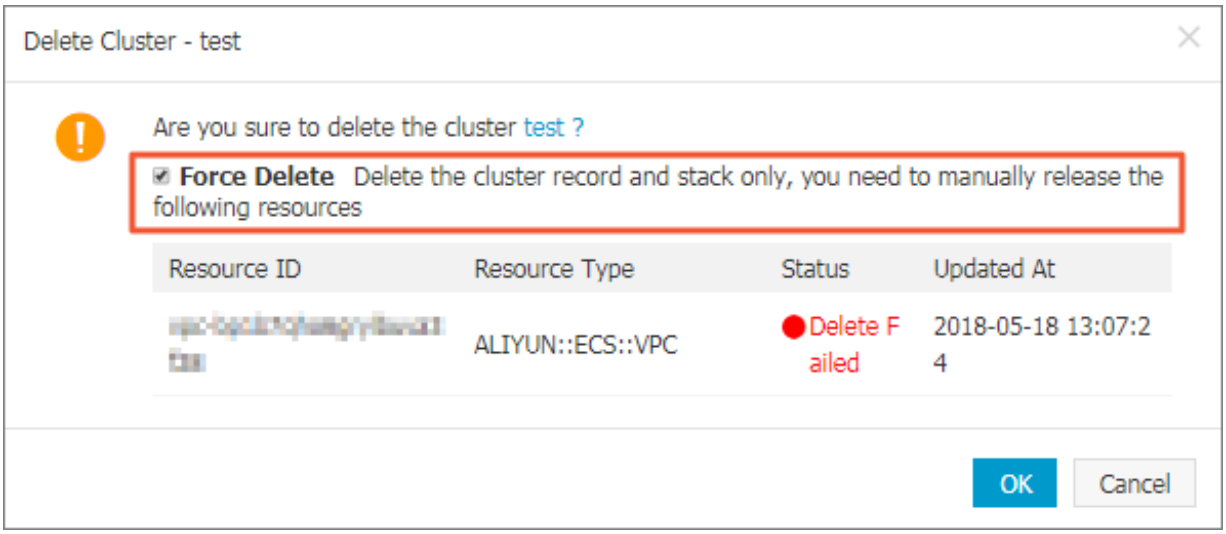
The cluster status is Failed to delete if the cluster fails to be deleted.



Click**More at the right of the cluster and then select** > **Delete.**In the displayed dialog box, you can see the resources that failed to be deleted. Select the **Force Delete** check box and then click **OK** to delete the cluster and ROS stack.

Note:

You must manually release the resources that failed to be deleted. To find these resources, see *Failed to delete Kubernetes clusters: ROS stack cannot be deleted*.

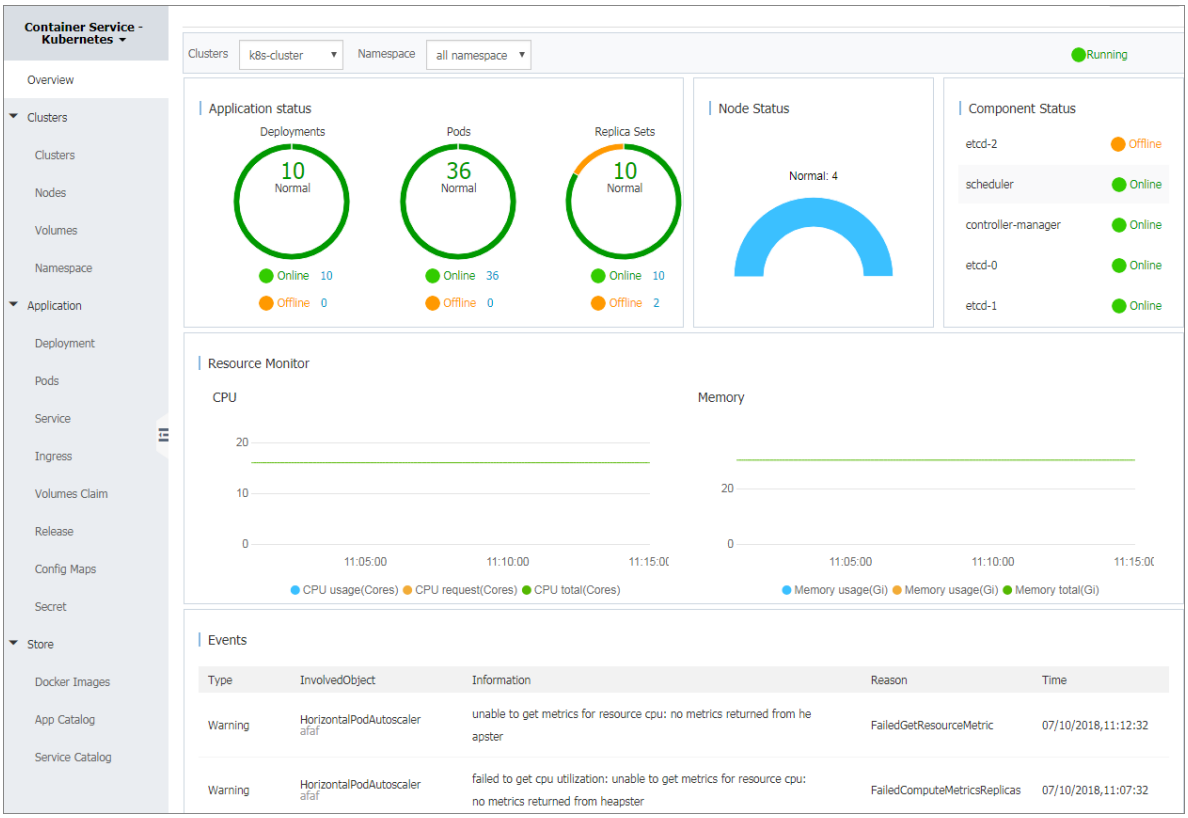


1.3.9 View cluster overview

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.



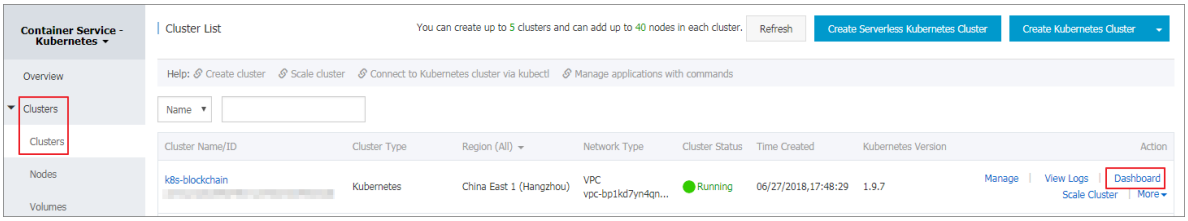
1.4 Application Management

1.4.1 Create an application in Kubernetes dashboard

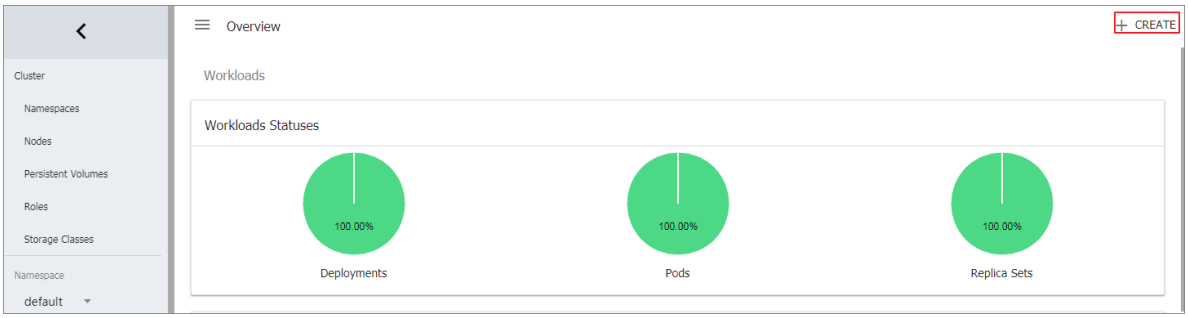
You can create an application in the Kubernetes dashboard.

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.

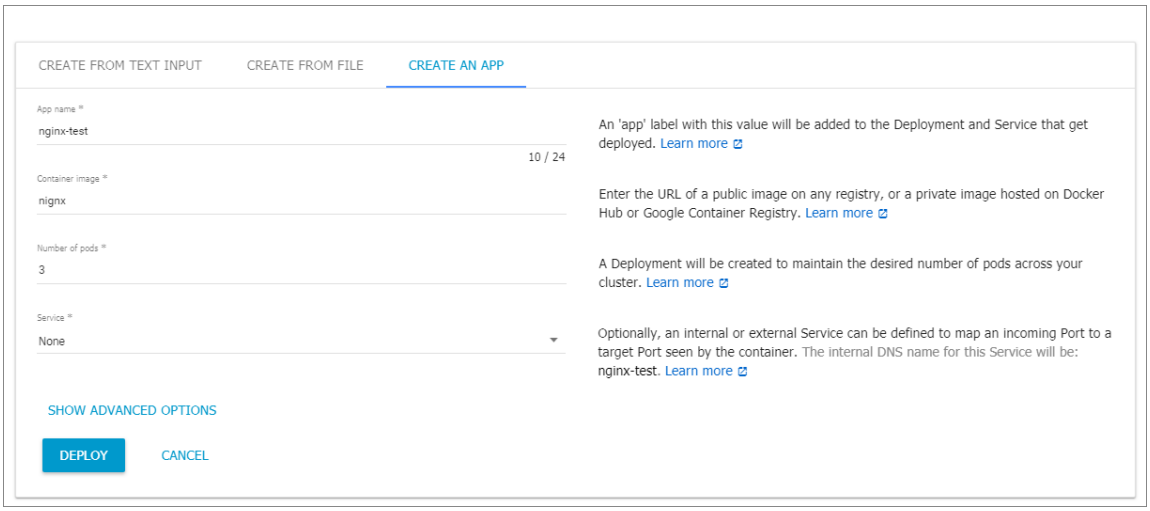


5. The Resource creation page appears. Configure the application information.

   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.

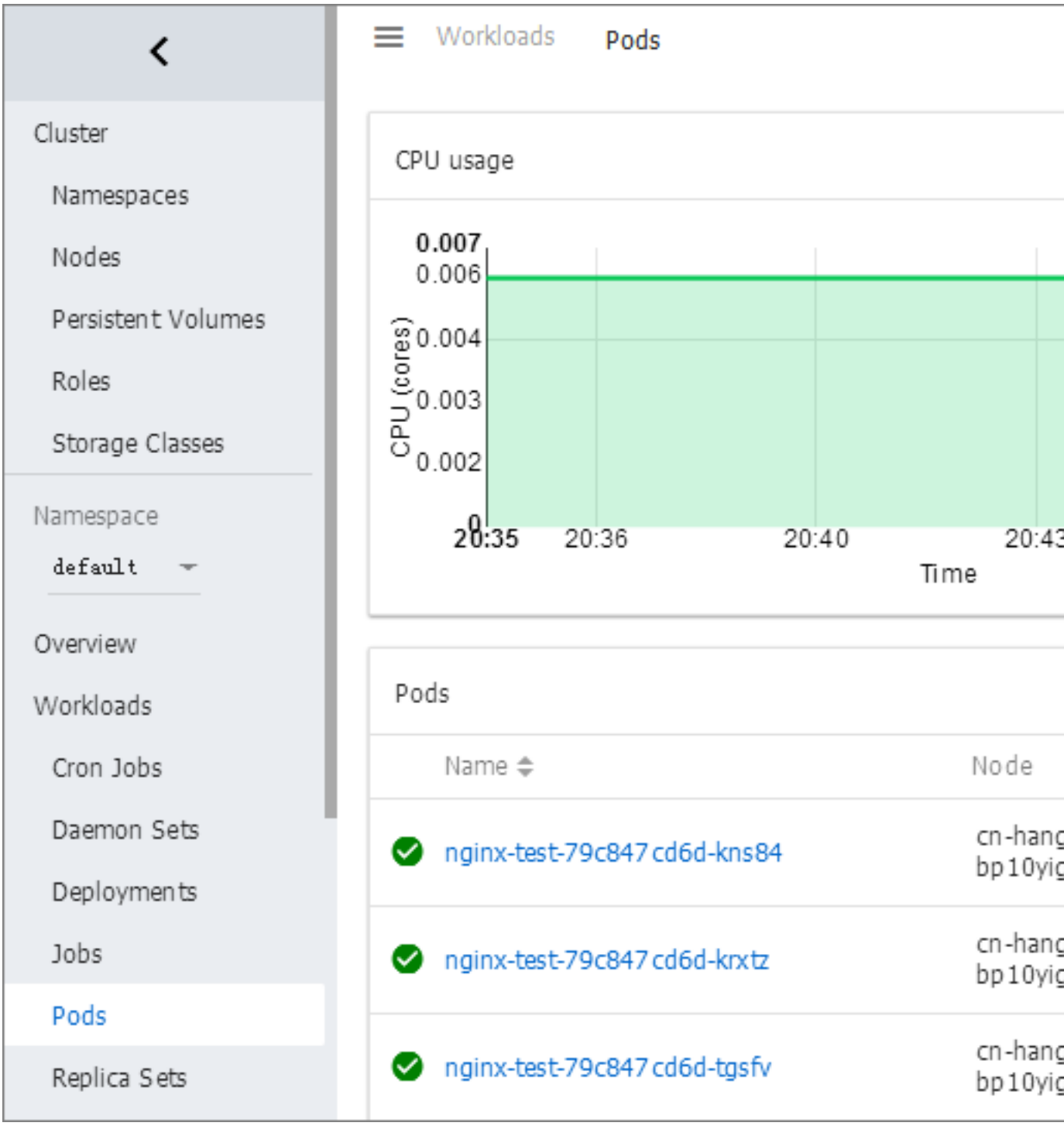
6. Click UPLOAD or **DEPLOY** to deploy the containers and services.

   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.



1.4.2 Create an application by using an image

Prerequisites

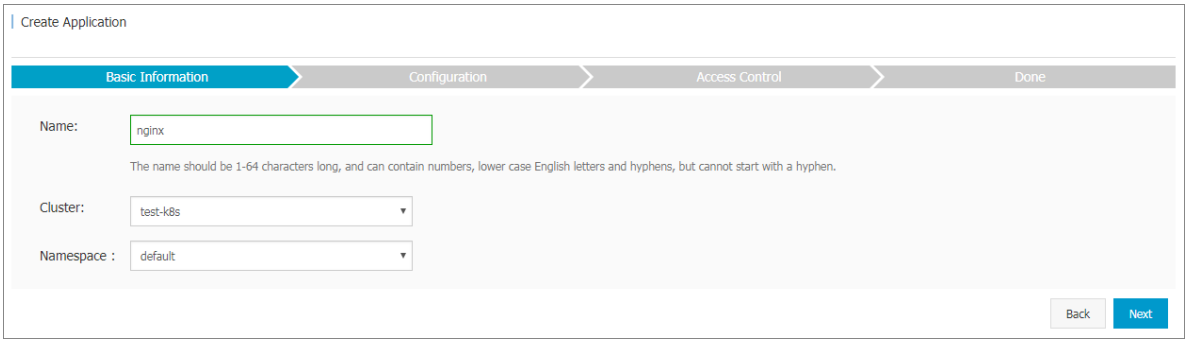
Create a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

Procedure

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

2. Click Kubernetes >**Application >** > **Deployment in the left-side navigation pane.**Click **Create by image** in the upper-right corner.
3. Enter the application **Name**, and then select the **Cluster** and **Namespace**. Click **Next** to go to the Configuration step.

   By default, the system uses the default namespace if the **namespace** is not configured.



4. Configure the general settings for the application.

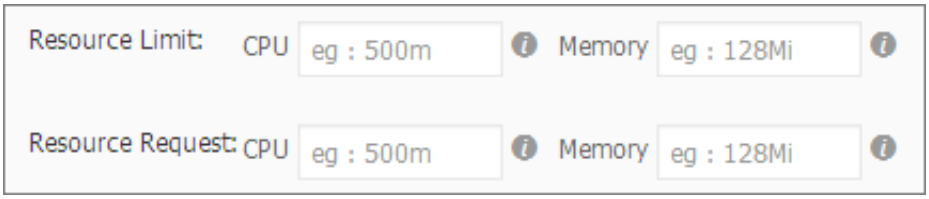
   - **Image Name**: You can click **Select image** to select the image in the displayed dialog box and then click **OK**. In this example, the image name is nginx.

     You can also enter the private registry in the format of `domainname/namespace/imagename:tag`.
   - **Image Version**: Click **Select image version** to select the image version. If the image version is not specified, the system uses the latest version by default.
   - **Scale**: Specify the number of containers. In this example, only one container is in the pod. If multiple containers are specified, the same number of pods will be started.

5. Configure the resource limit and resource reserve for the container.

   - **Resource Limit**: Specify the upper limit for the resources (CPU and memory) that can be used by this application to avoid occupying excessive resources.
   - **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.

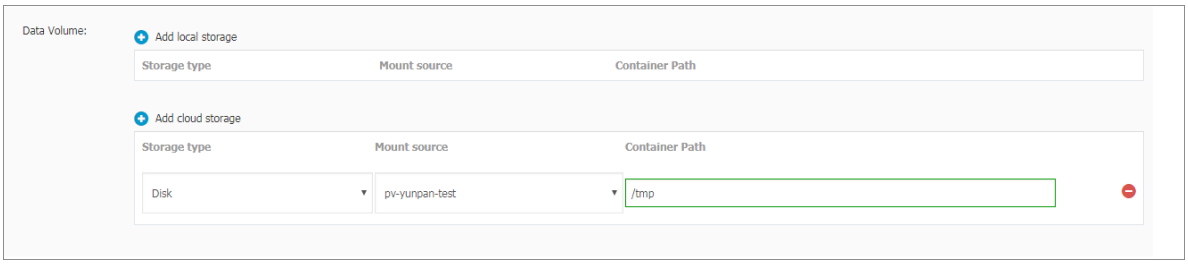
CPU is measured in millicores (one thousandth of one core). Memory is measured in bytes, which can be Gi, Mi, or Ki.



6. Configure the data volumes.

   The hostPath data volumes can be configured. The hostPath data volumes mount the files or directories in the host file system to the pod. For more information, see hostPath in *volumes*.

   In this example, configure a hostPath data volume named data. Use the host directory `/tmp` and mount the data volume data to the `var/lib/docker` directory in the container.



7. Configure the environment variable.

   You can configure the environment variable for the pod in the format of key-value pairs to add the environment label or pass the configurations for the pod. For more information, see *Pod variable*.

8. Configure the container.

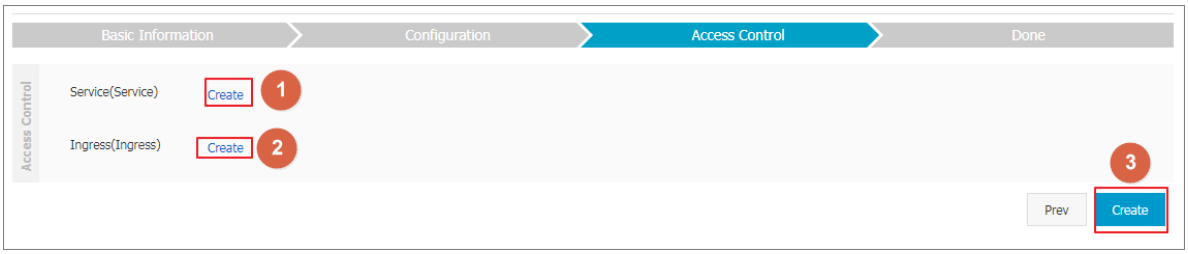
   You can configure the Command, Args, and Container Config for the container running in the pod.

   - **Command** and **Args**: If not configured, the default settings of the image are used. If configured, the default settings of the image are overwritten. If only the Args is configured, the default command will run the new arguments when the container is started. Command and Args cannot be modified after the pod is created.
   - **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 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.

9. Select whether or not to enable the **Auto Scaling**.

   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.

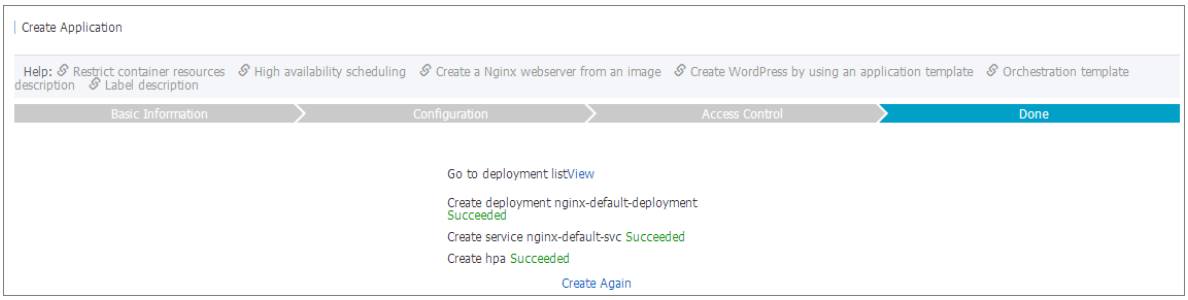
10. Click **Next** after completing the configurations. In the Access Control step, configure a service to bind with the backend pods.



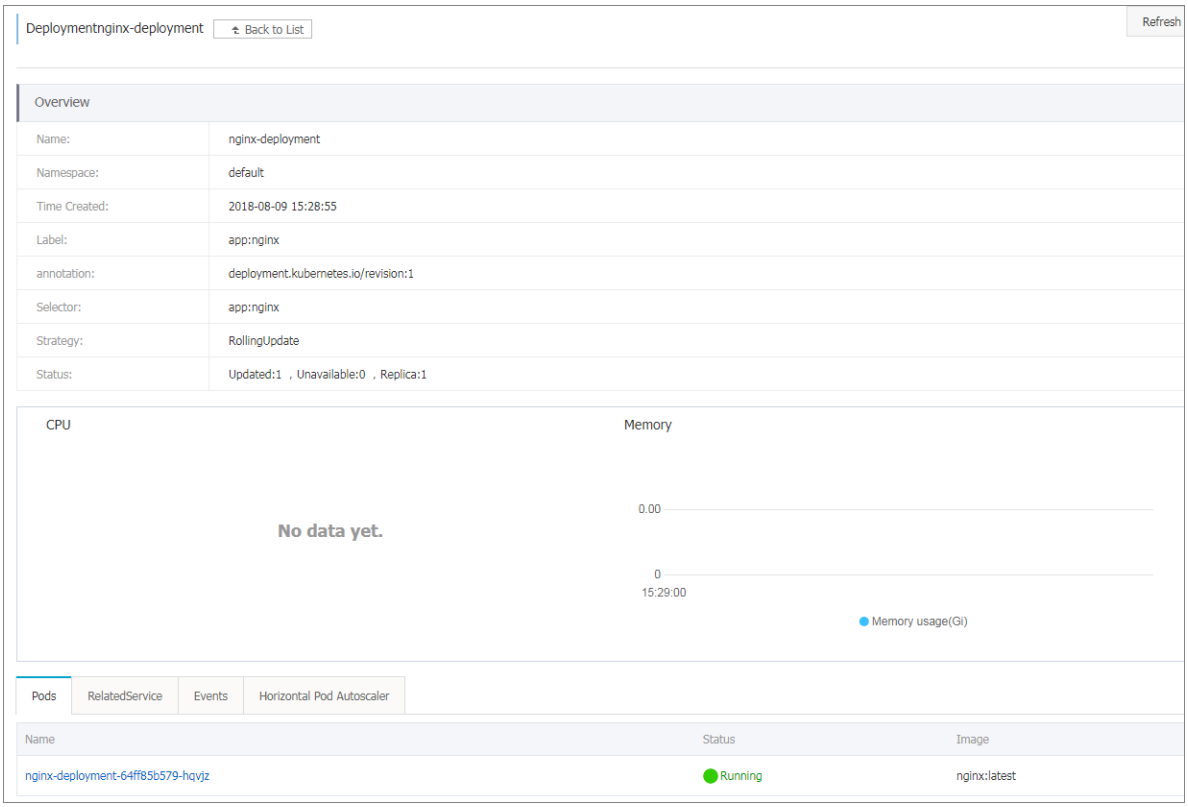
- **Service**: Select None to not create a service, or select a service type as follows:
  - ClusterIP: Exposes the service by using the internal IP address of your cluster. With this type selected, the service is only accessible from within the cluster.
  - NodePort: Exposes the service by using the 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. Server Load Balancer can route to the NodePort and ClusterIP services.
- **Name**: By default, a service name composed of the application name and the suffix svc is generated. In this example, the generated service name is nginx-default-svc. You can modify the service name as needed.
- **Port Mapping**: You must add the service port and the container port. If NodePort is selected as the service type, you must configure the node port to avoid the port conflict. Select TCP or UDP as the Protocol.

11. Click **Create** after the access control configurations.

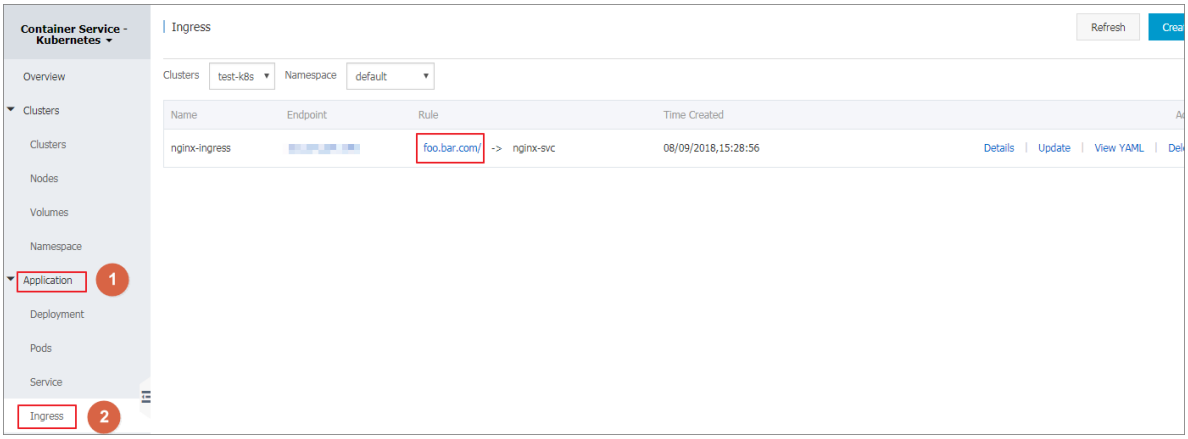
    The Done step indicating the successful creation appears. The objects contained in the application are displayed. You can click View to view the deployment list.



12. The newly created deployment nginx-default-deployment is displayed on the Deployment page.



13. Click**Application >** > **Service**in the left-side navigation pane. The newly created service nginx-default-svc is displayed on the Service List page.



14.Access the external endpoint in the browser to access the Nginx welcome page.

1.4.3 Create an application by using an orchestration template

Prerequisites

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

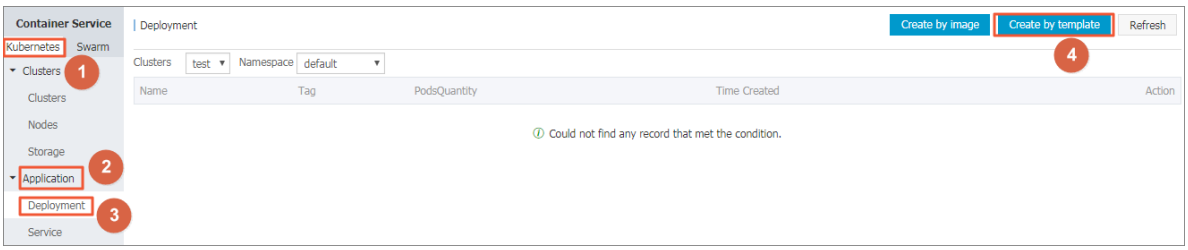
Context

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

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.

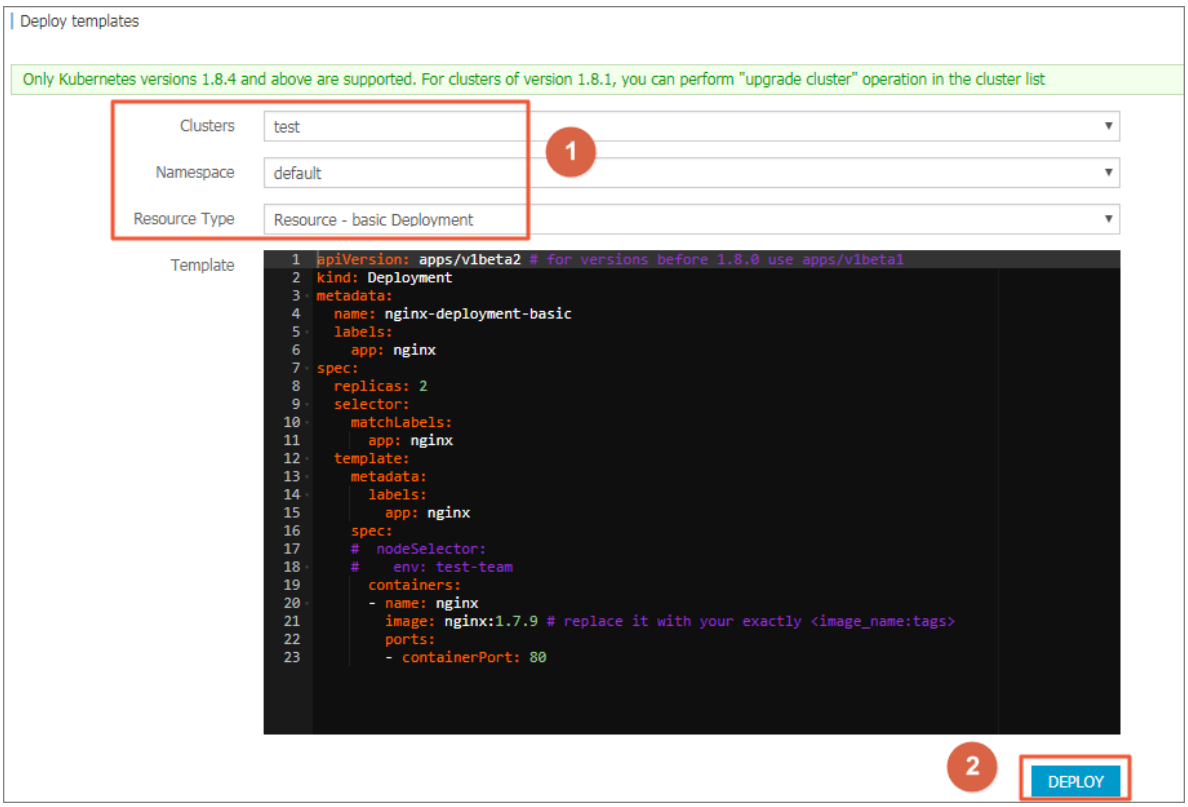
Procedure

1. Log on to the *Container Service console*.
2. Click Kubernetes >**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 the resource object is to be deployed.

- **Namespace:** Select the namespace to which the resource object belongs. default is selected by 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.

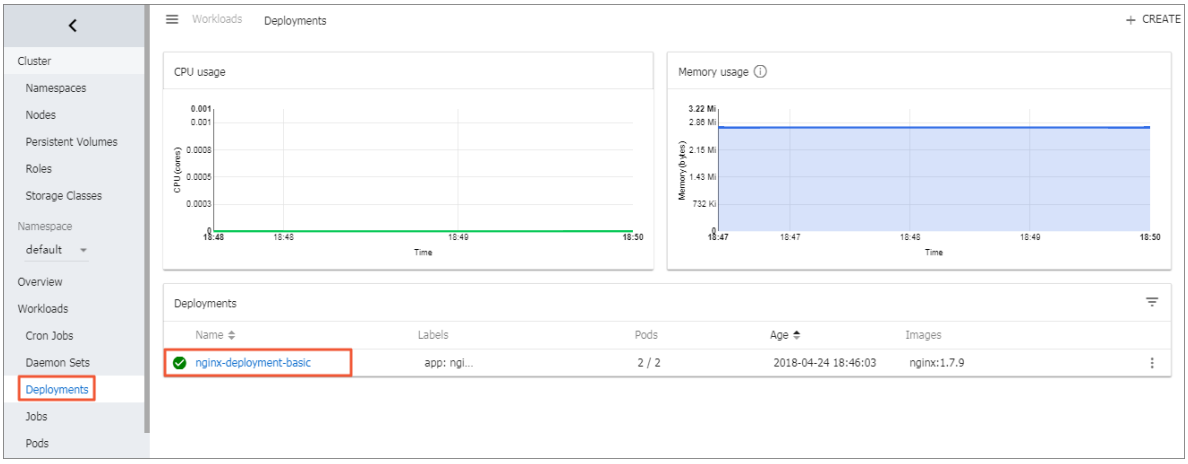


The deployment sample orchestration of an Nginx application is as follows. By using this orchestration template, you can create a deployment that belongs to an Nginx application quickly.

```
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
```

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



6. Go back to the **Deploy templates** page and deploy a service resource object.

Container Service provides a service sample template of an Nginx application. In this example , modify the template a little by changing the access type to LoadBalancer, and then you can create a service bound to the backend pod, which allows you to access the service in the browser.

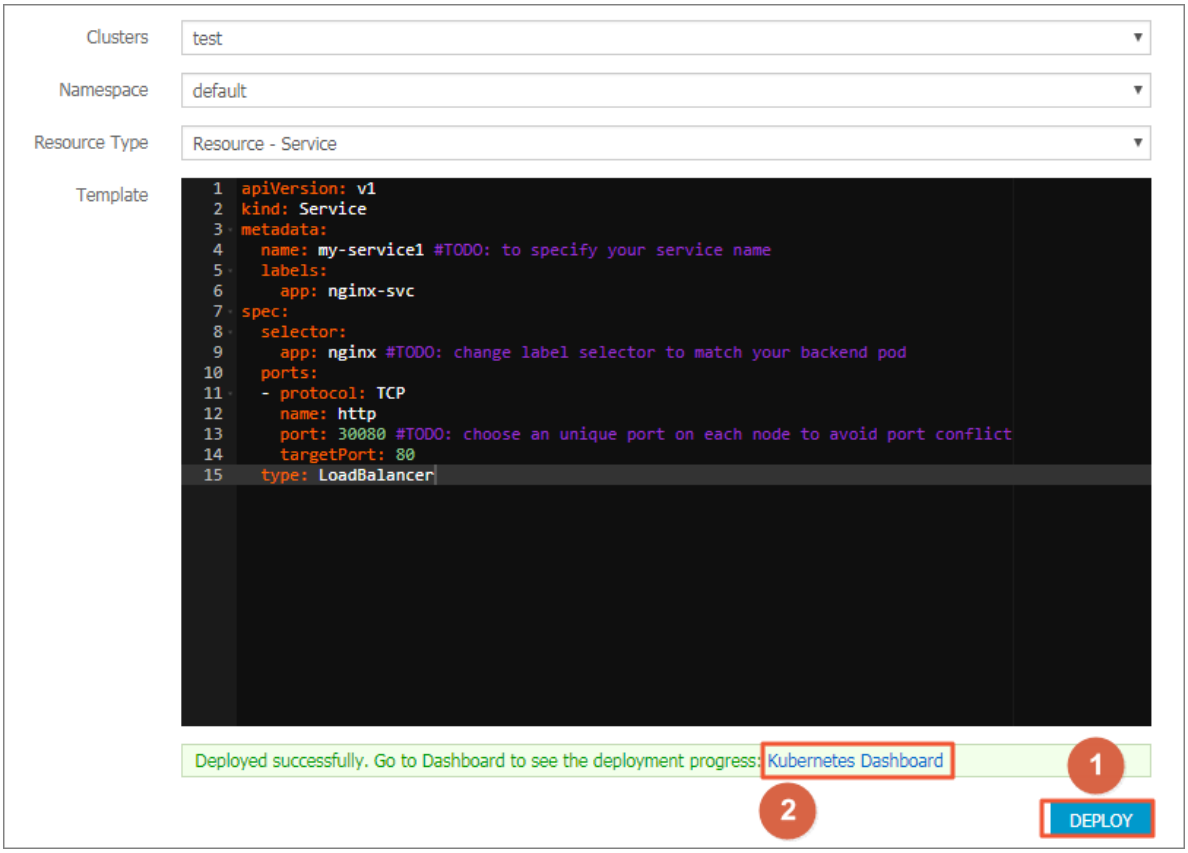
Note:

In this example, the selector values in the pod orchestration and service orchestration are both nginx, so no modification is required. Make the corresponding modifications according to your actual situations.

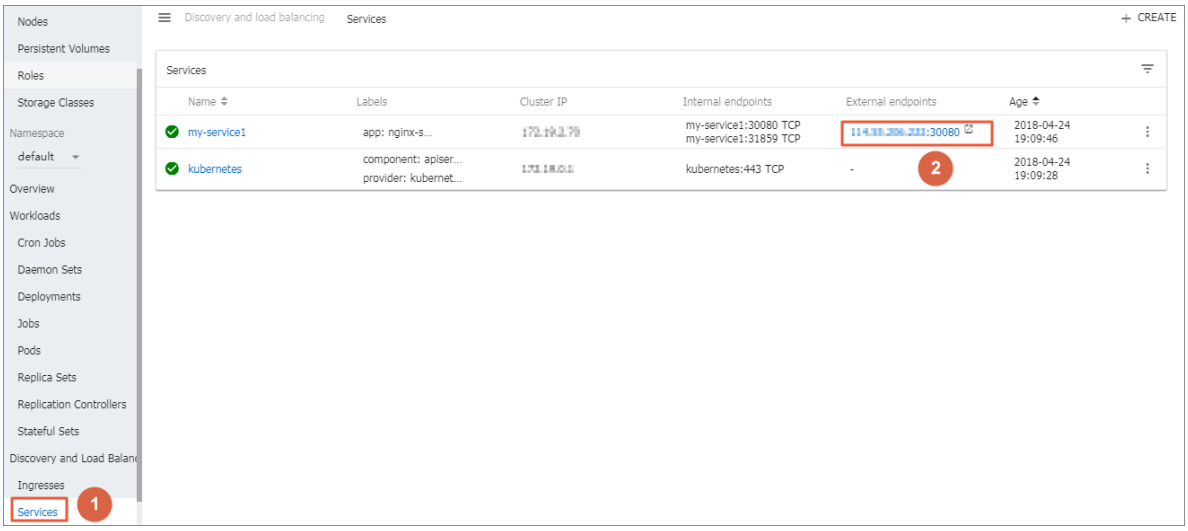
```
apiVersion: v1 # for versions before 1.8.0 use apps/v1beta1
 kind: Service
 metadata:
   name: my-service1 #TODO: to specify your service name
   labels:
     app: nginx
 spec:
   selector:
     app: nginx #TODO: change label selector to match your backend
pod
   ports:
```

```
    - 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.
```

7. Enter the preceding orchestration contents in the Template field and then click **DEPLOY**. 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 progress of the service.



8. In the Kubernetes dashboard, you can see the service my-service1 is successfully deployed and exposes the external endpoint. Click the access address under **External endpoints**.



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

1.4.4 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*.

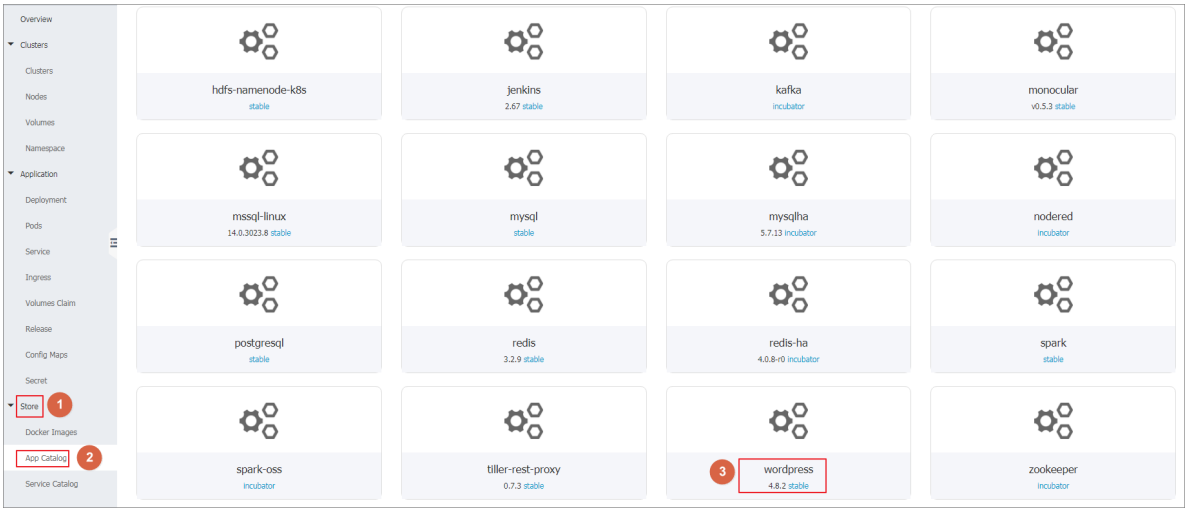
  Tiller is automatically deployed to the cluster when the Kubernetes cluster is created. Helm CLI is automatically installed on all the master nodes and the configuration points to the Alibaba Cloud chart repository.
- Check the Kubernetes version of your cluster.

  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

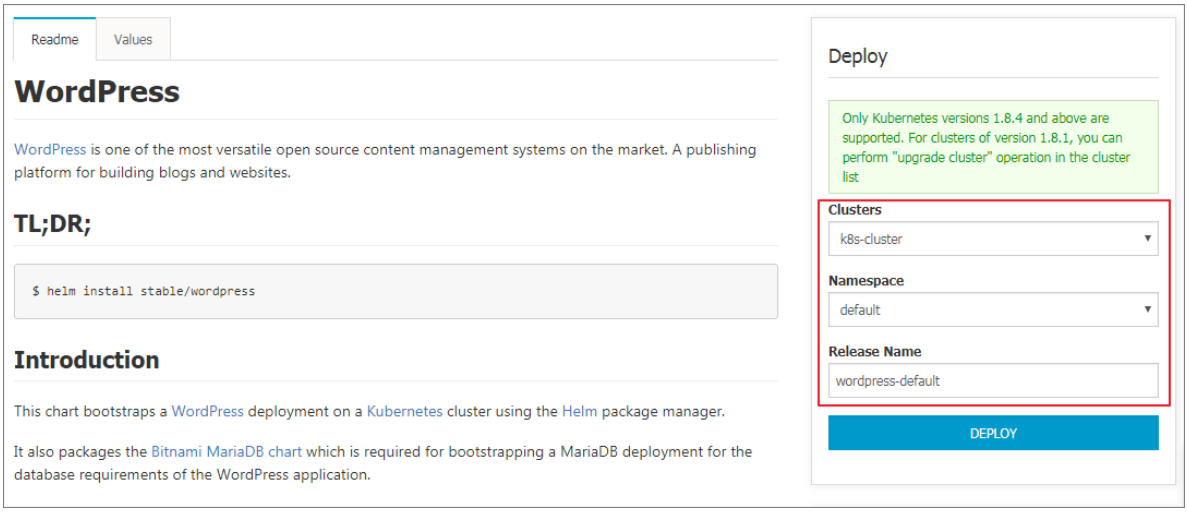
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Store** > **App Catalog** in the left-side navigation pane.

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.

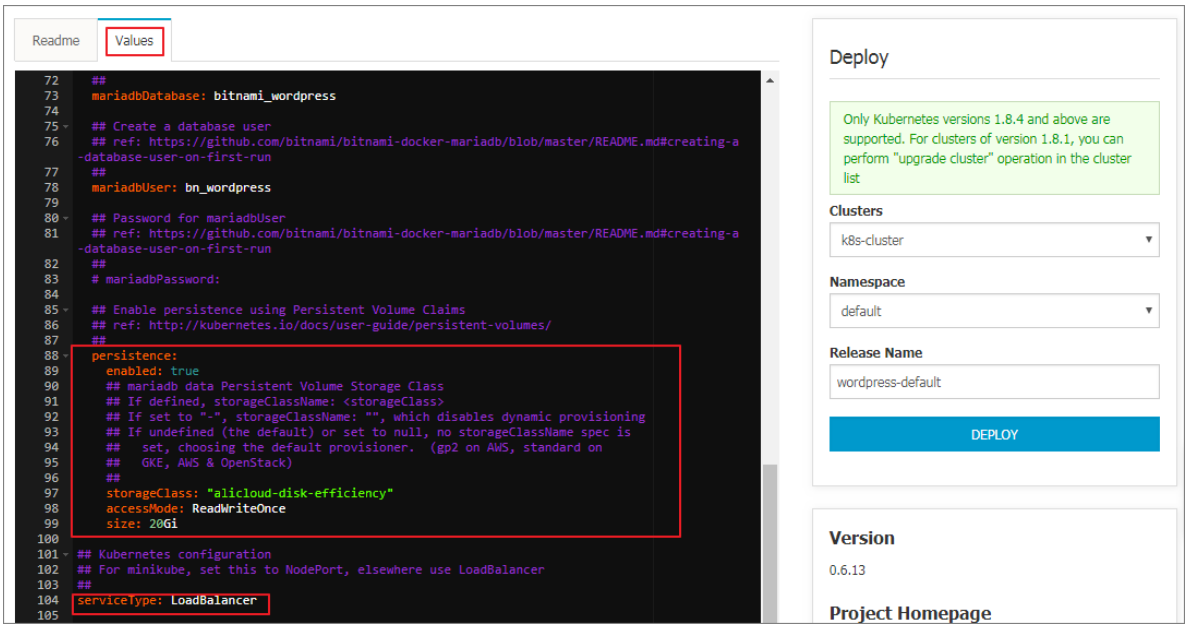
   - **Clusters**: Select the cluster in which the application is to be deployed.
   - **Namespace**: Select the namespace. default is selected by default.
   - **Release Name**: Enter the release name for the application. Enter test in this example.



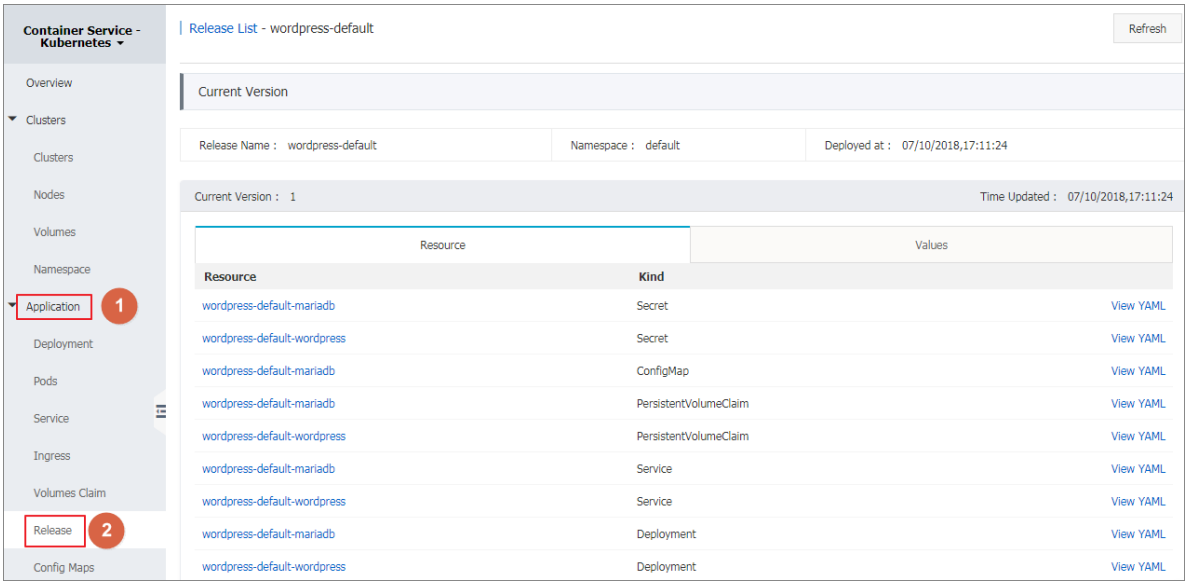
5. Click the **Values** tab to modify the configurations.

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

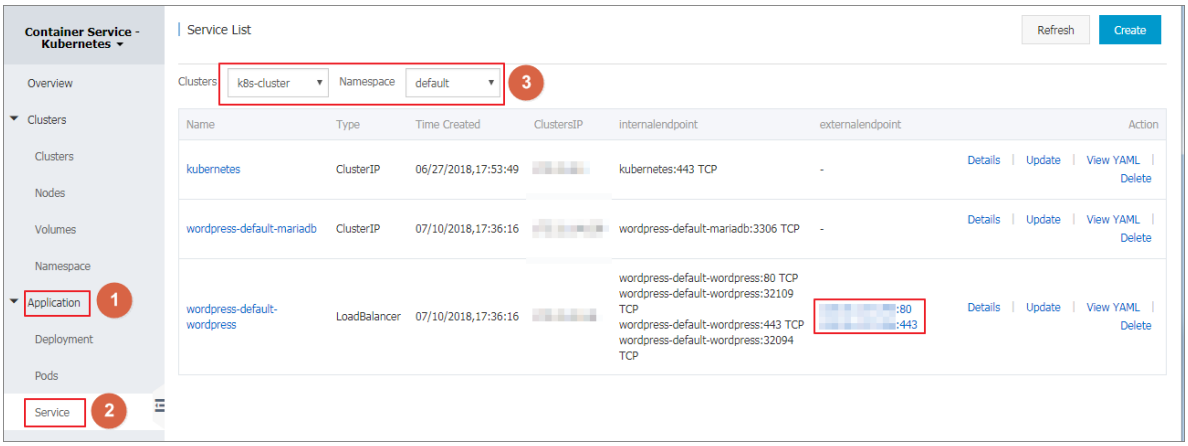
> **Note:**
> You need to create a persistent storage volume (PV) of cloud disk in advance. The capacity of the PV cannot be less than the value defined by the PVC.



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/`.

1.4.5 Manage applications by using commands

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

Prerequisites

Before using commands to manage applications, *Connect to a Kubernetes cluster by using kubectl*.

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
```

1.4.6 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*.

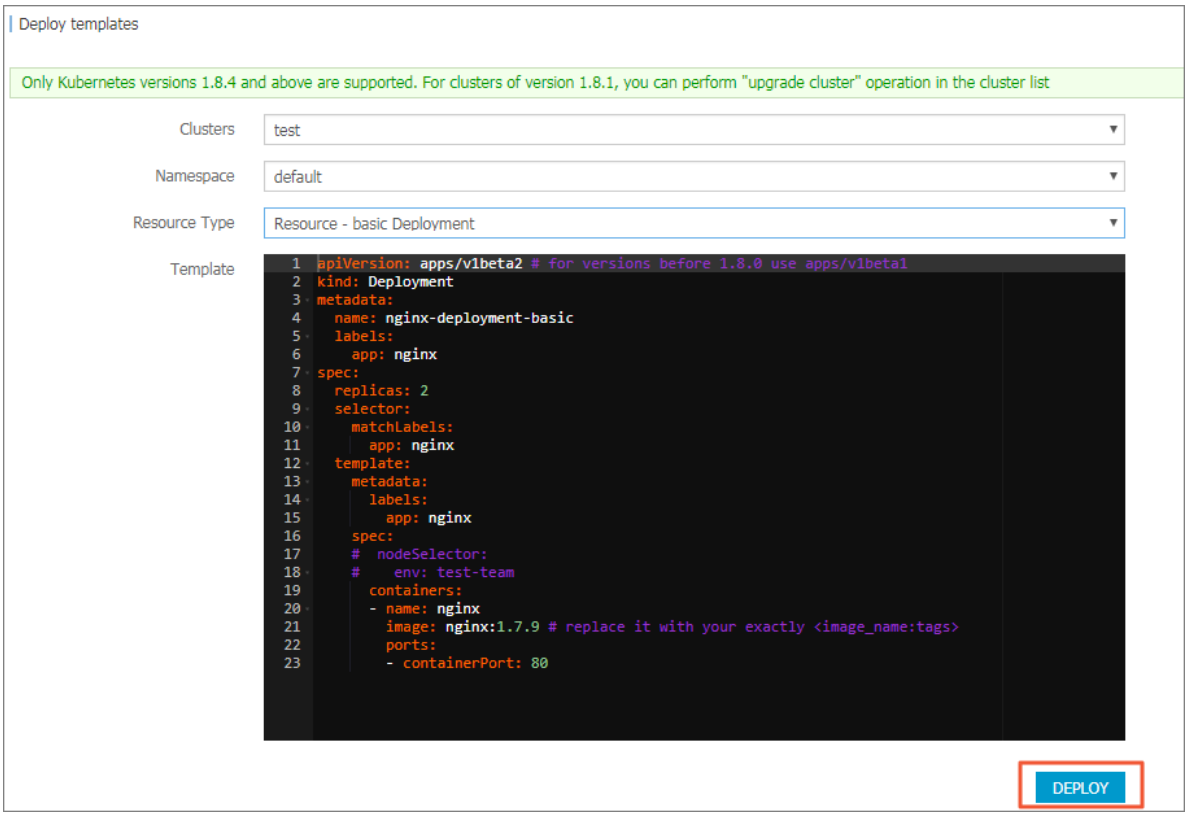
Prerequisites

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

Step 1 Create a deployment

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

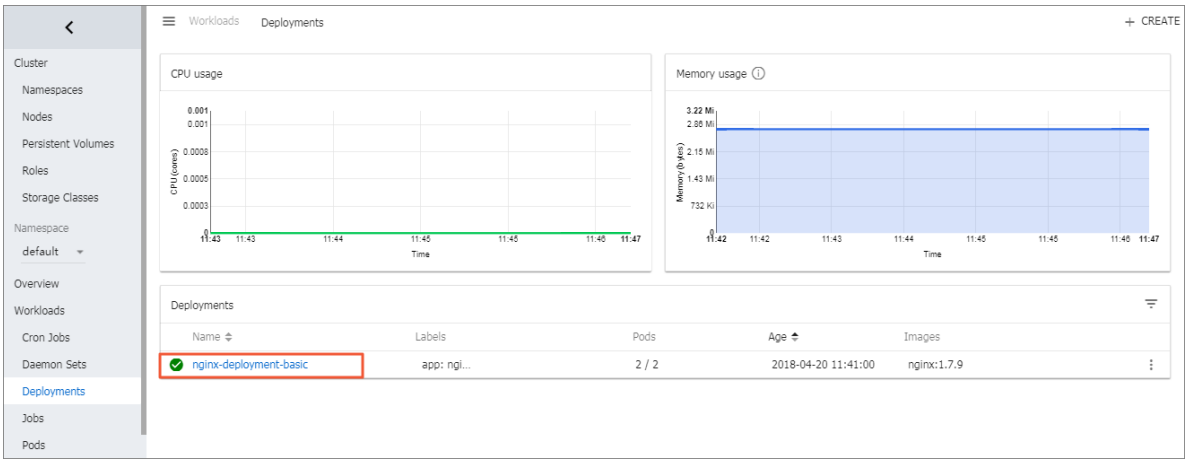
4. Select the cluster and namespace to create the deployment. In the Resource Type drop-down list, select Custom to customize the template or a sample template. Then, click **DEPLOY**.



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

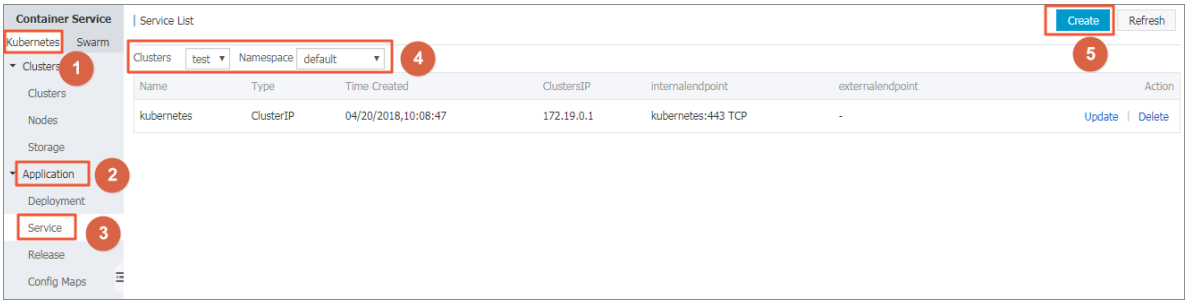
```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/
v1beta1
 kind: Deployment
 metadata:
   name: nginx-deployment-basic
   labels:
     app: nginx
 spec:
   replicas: 2
   selector:
     matchLabels:
       app: nginx
   template:
     metadata:
       labels:
         app: nginx
   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

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

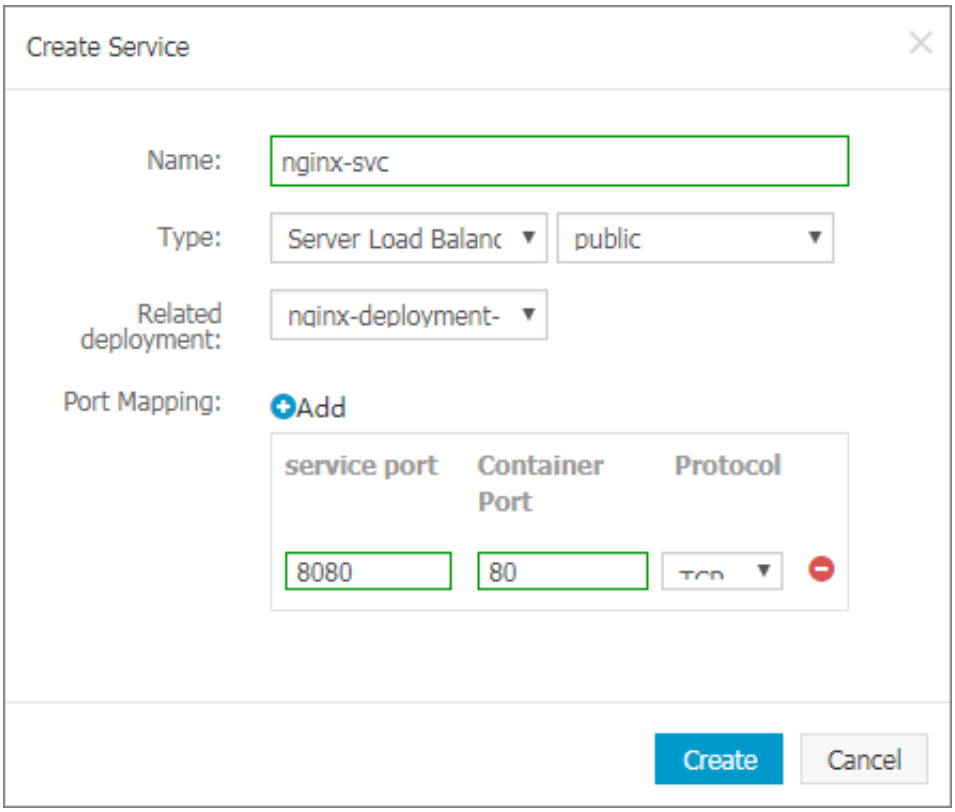


Step 2 Create a service

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



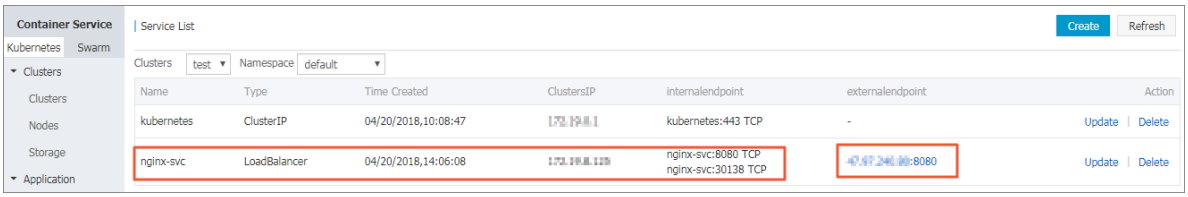
5. Complete the configurations in the displayed Create Service dialog box.



- **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.

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



7. 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.

1.4.7 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*.

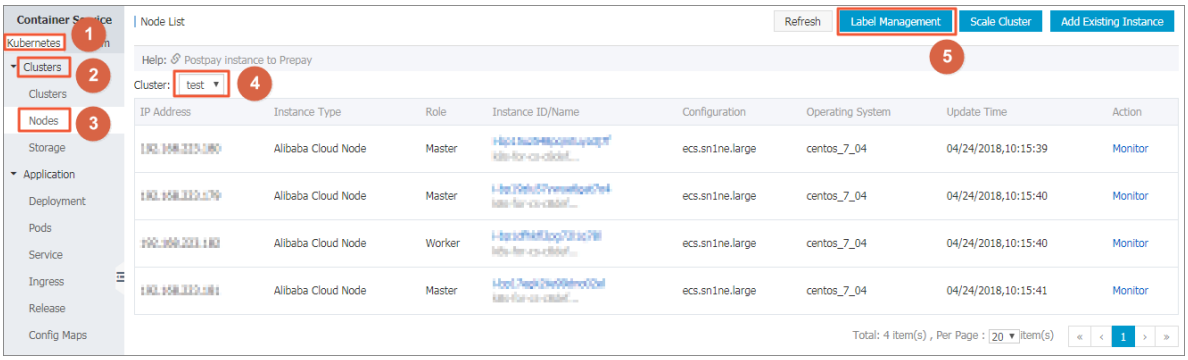
For business scenario needs, to deploy a service used for management and control to a master node, or deploy services to a machine with an SSD disk, you can use this method to schedule pods to specified nodes.

Prerequisites

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

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.

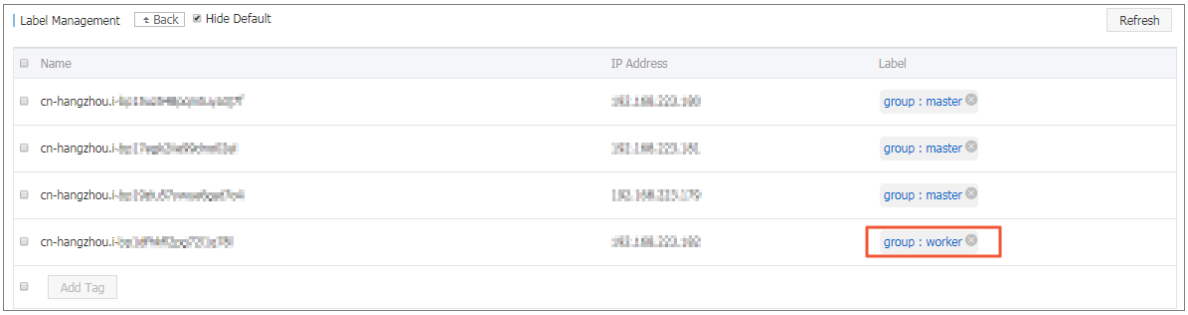


4. Select one or more nodes by selecting the corresponding check boxes and then click **Add Tag**.

   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.

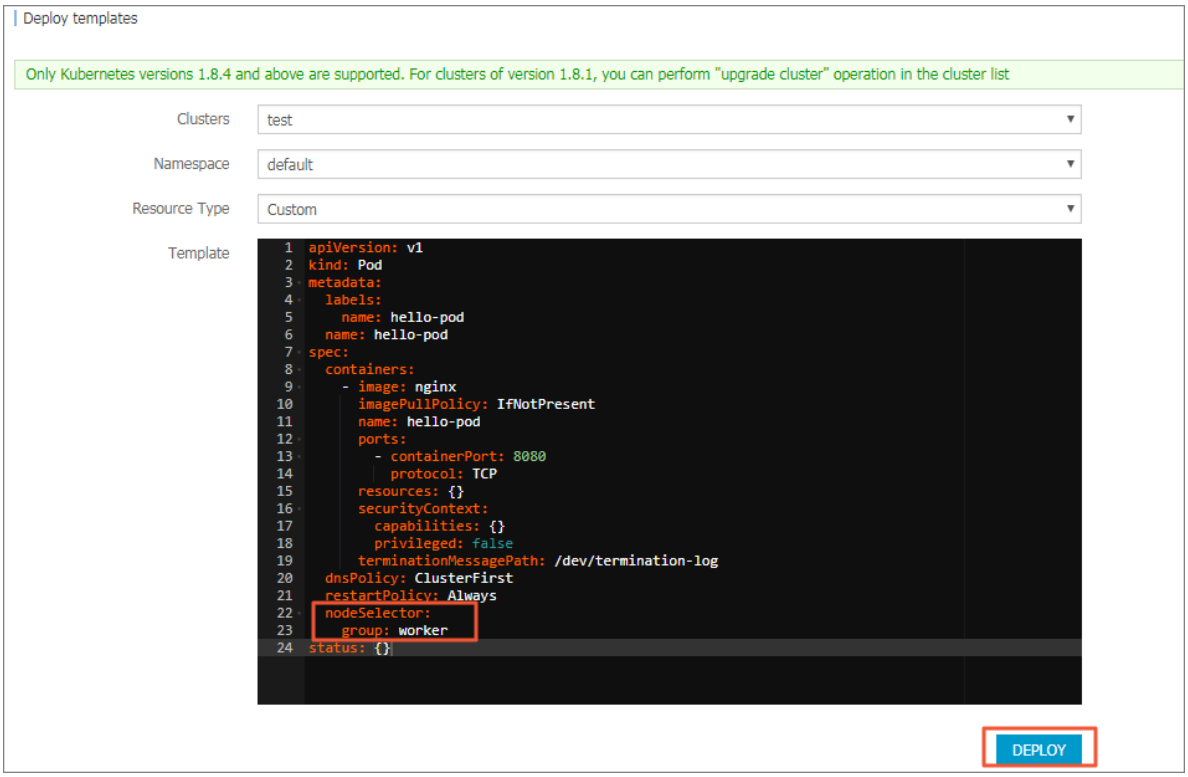


You can also add a node label by running the command `kubectl label nodes <node-name> <label-key>=<label-value>`.

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.

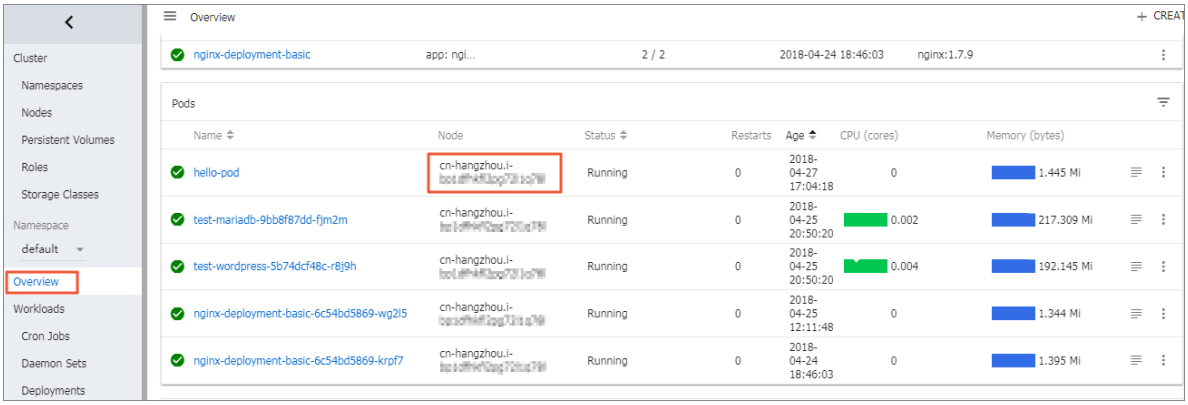


The orchestration template in this example is as follows:

```
apiVersion: v1
```

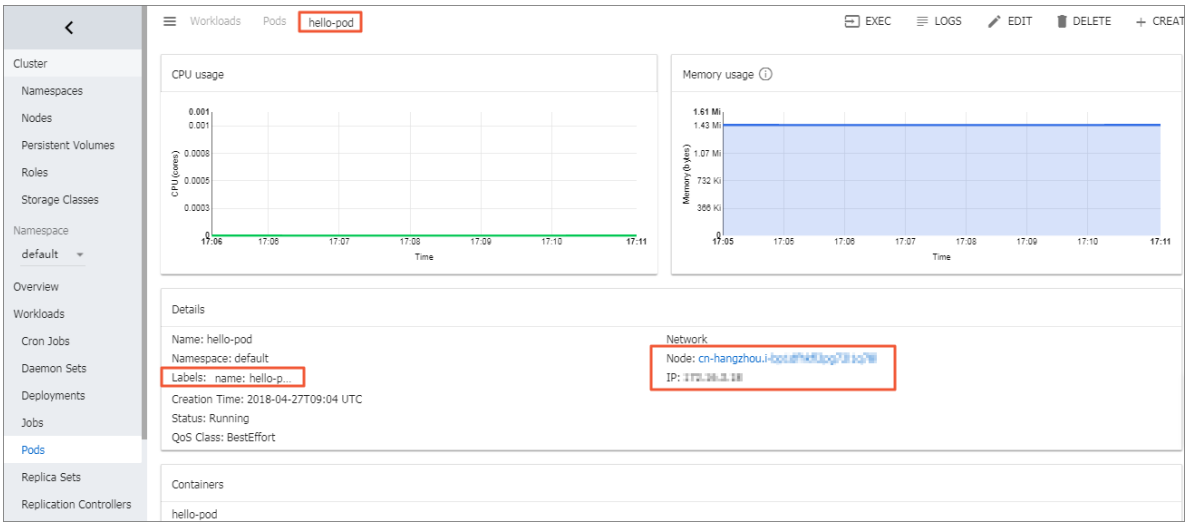
```
 kind: Pod
 metadata:
   labels:
     name: hello-pod
   name: hello-pod
 spec:
   containers:
     - image: nginx
       imagePullPolicy: IfNotPresent
       name: hello-pod
       ports:
         - containerPort: 8080
           protocol: TCP
       resources: {}
       securityContext:
         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`.

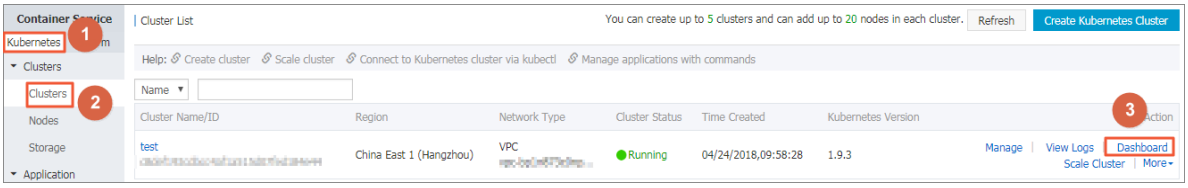


1.4.8 Service scaling

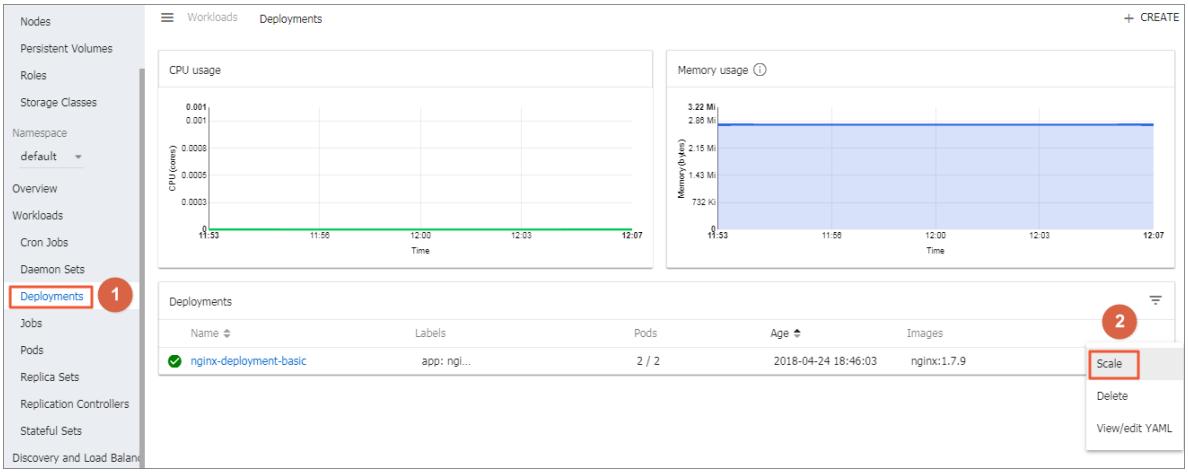
After an application is created, you can scale out or in the services as per your needs.

Procedure

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

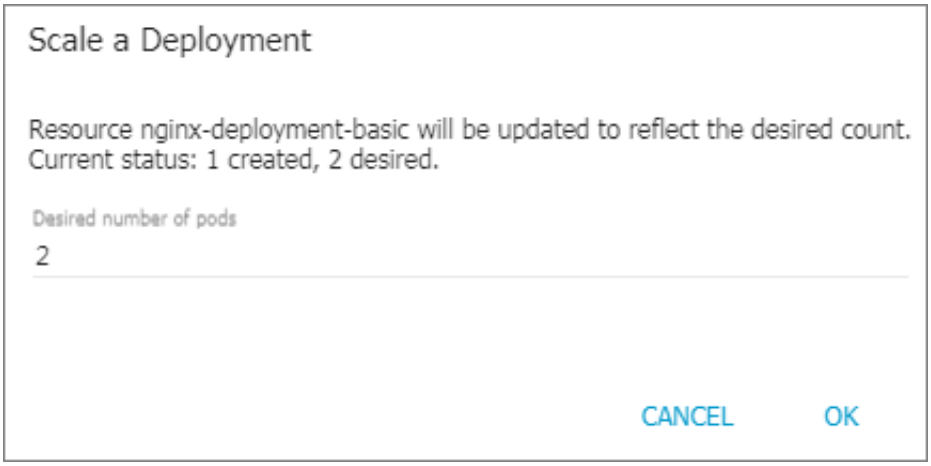


4. In the Kubernetes dashboard, click **Deployments** in the left-side navigation pane to view the created deployments.
5. Click the icon at the right of the deployment and then select **Scale**.



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

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

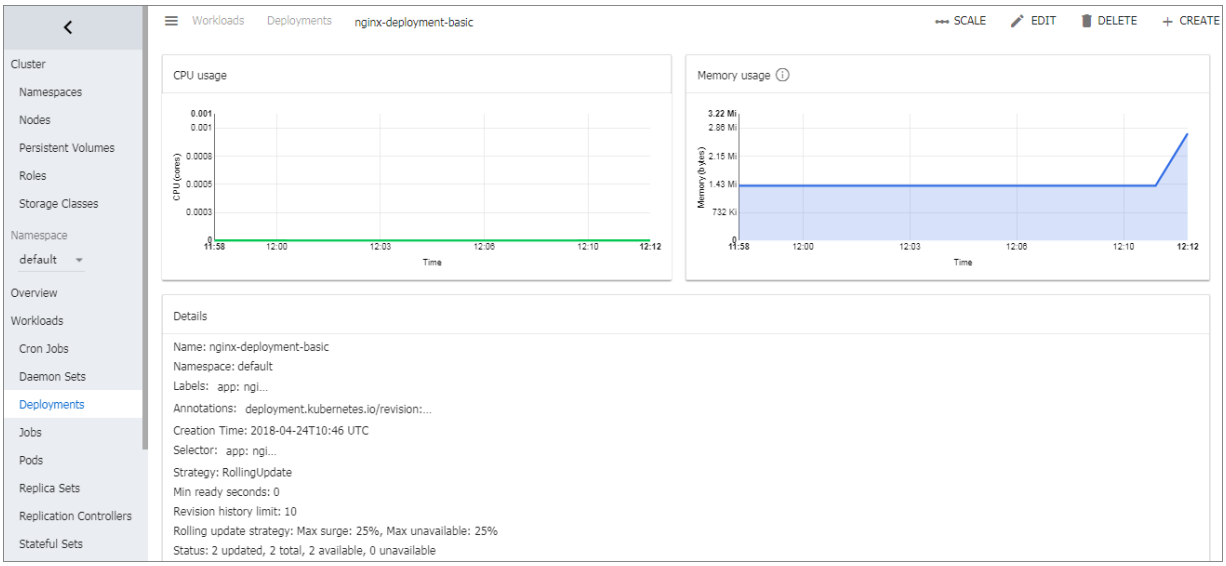


What's next

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

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

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



1.4.9 View services

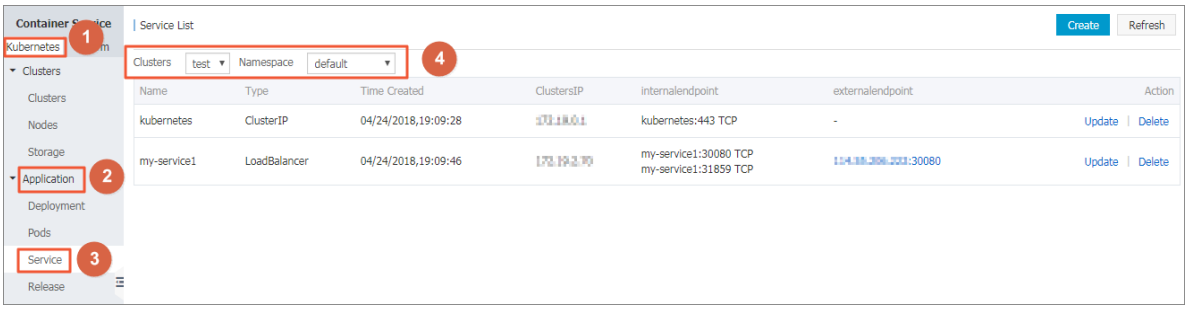
Context

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

Procedure

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

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



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

1.4.10 Delete a service

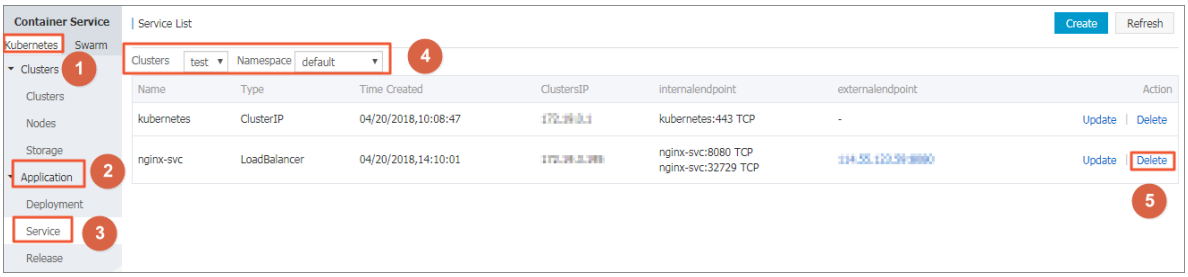
You can delete a Kubernetes service in the Container Service console.

Prerequisites

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

Procedure

1. Log on to the *Container Service console*.
2. Click Kubernetes >**Application >** > **Service**in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Delete** at the right of the service (nginx-svc in this example).



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

1.4.11 View pods

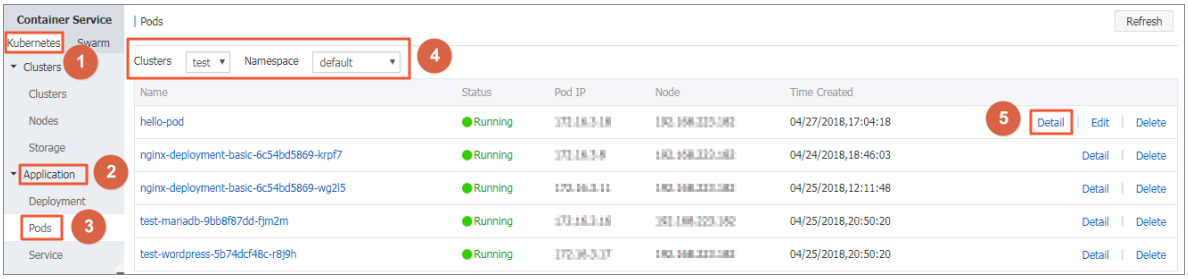
You can view the pods of a Kubernetes cluster in the Container Service console or in the Kubernetes dashboard.

View pods in Container Service console

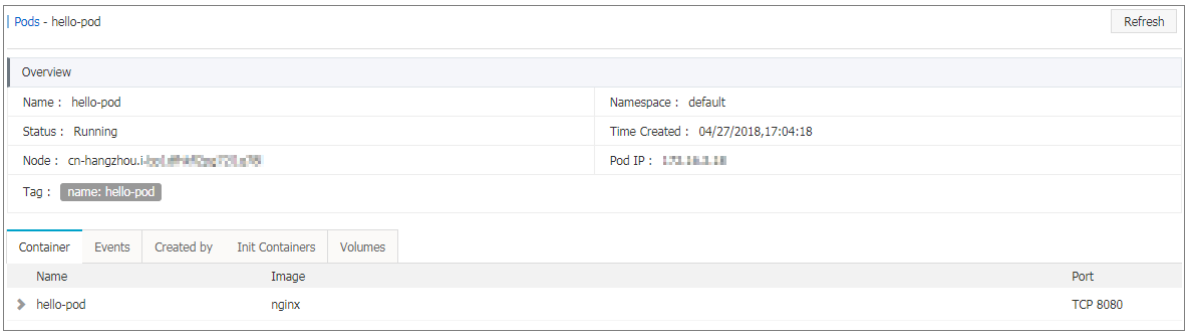
1. Log on to the *Container Service console*.
2. Log on to the *Container Service console*.

3. Click Kubernetes >**Application** > > **Pods**in the left-side navigation pane.
4. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Detail** at the right of the pod.

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

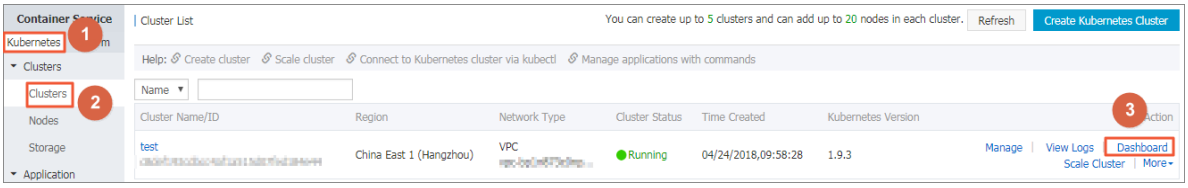


5. View the pod details.



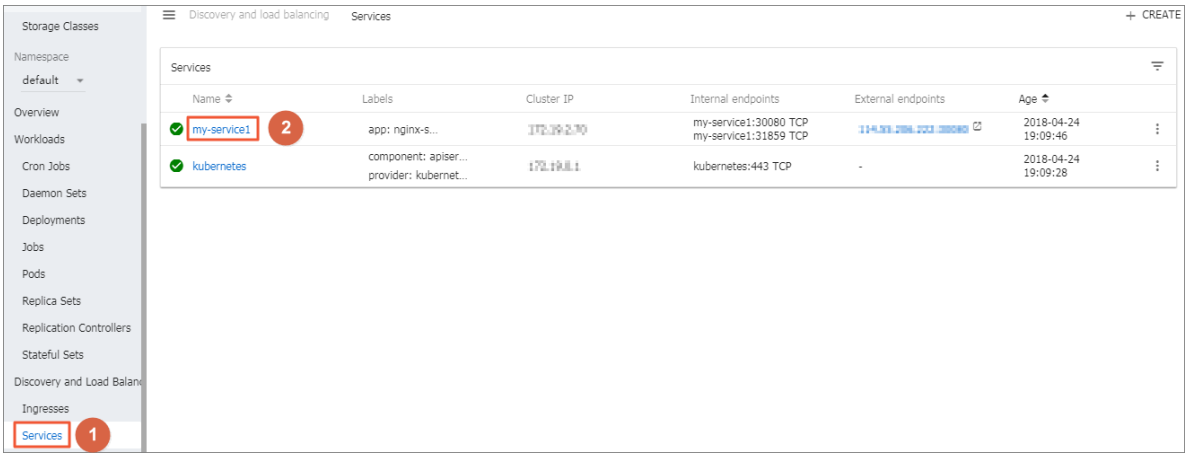
View pods in Kubernetes dashboard

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

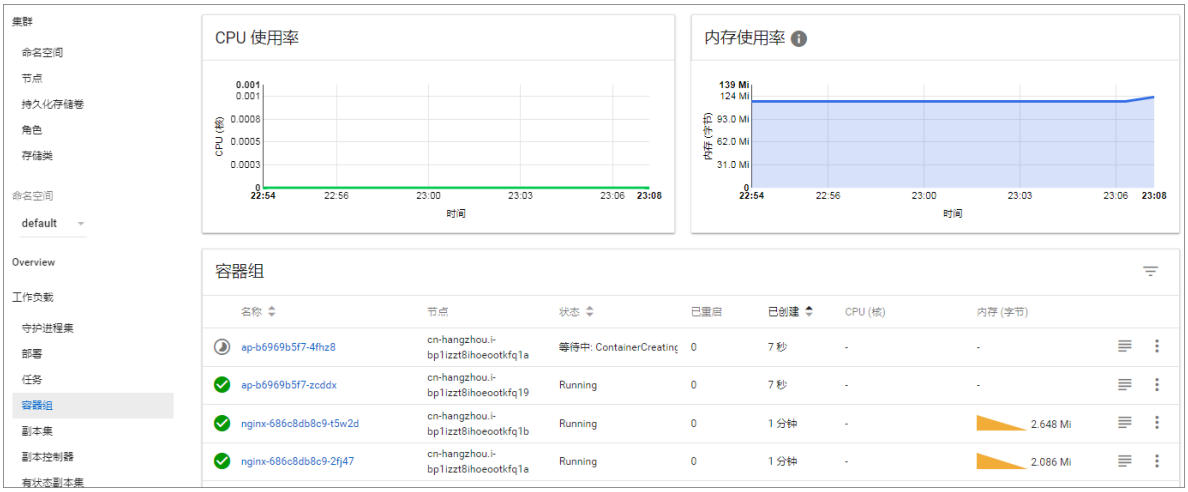


5. In the Kubernetes dashboard, click **Pods** in the left-side navigation pane to view the pods in the cluster.

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

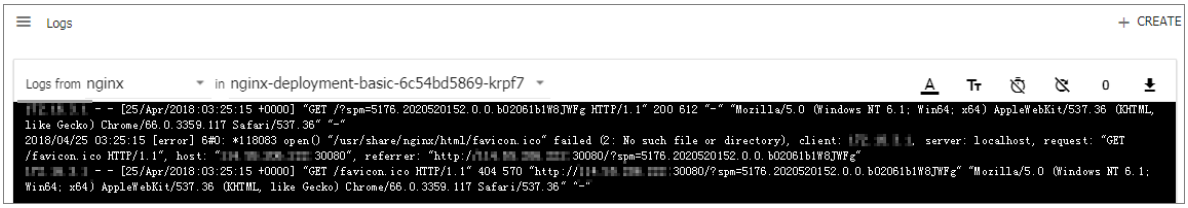


6. You can check the status of each Kubernetes object according to the icon on the left. indicates the object is being deployed. indicates the object has completed the deployment.

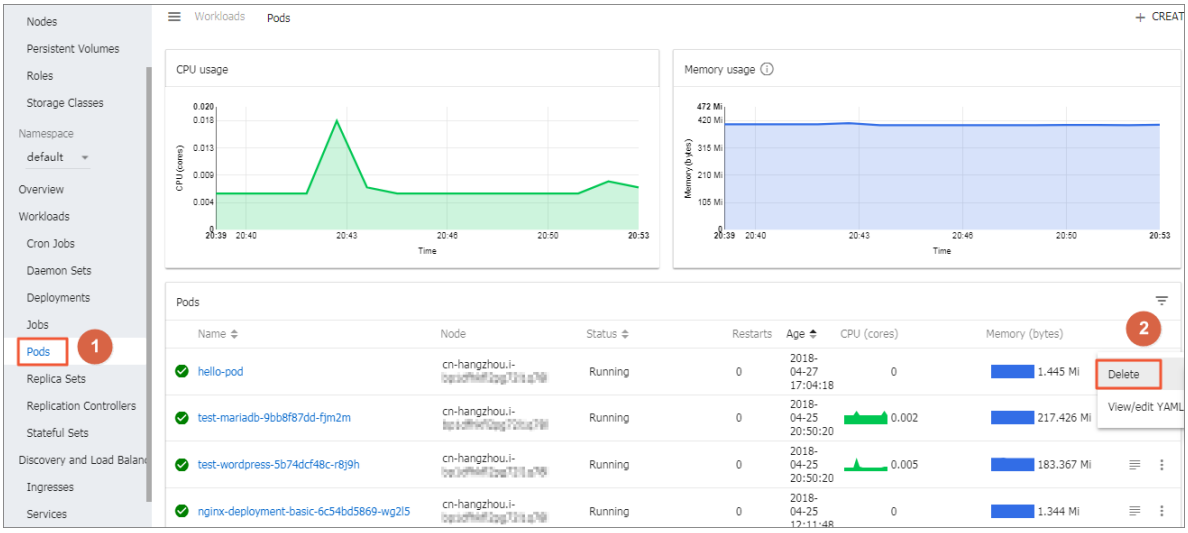


7. Click the pod name to view the details, CPU usage, and memory usage of the pod.

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



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

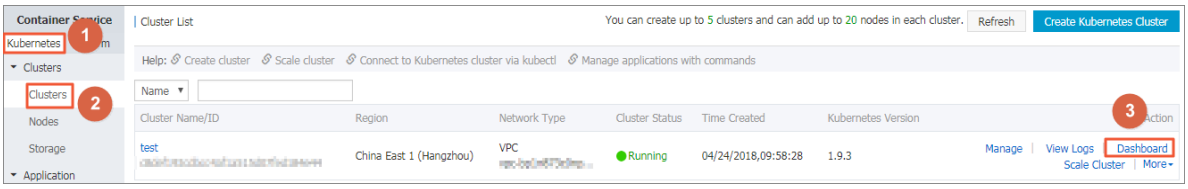


1.4.12 Change container configurations

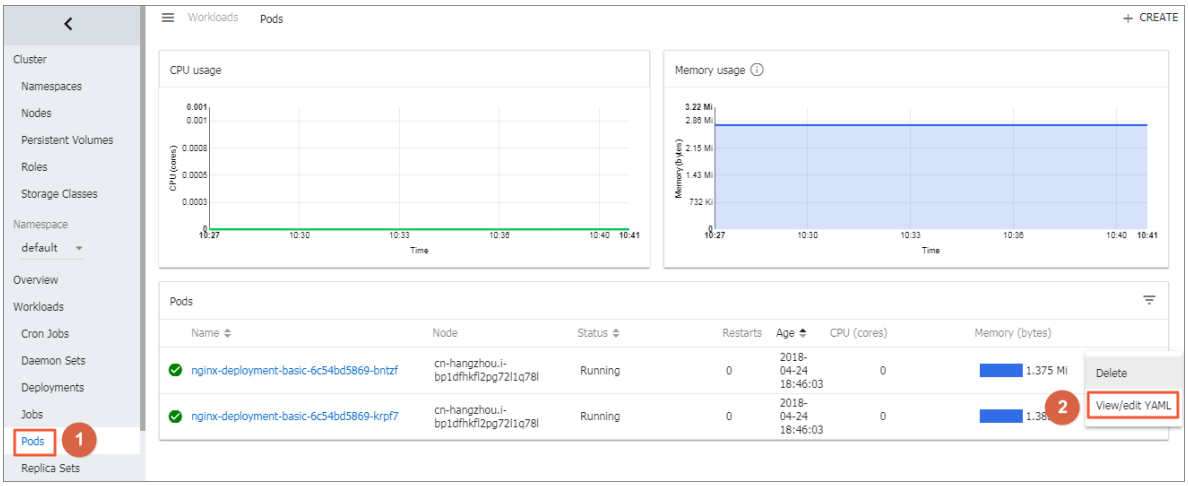
You can change the container configurations in the Container Service console.

Procedure

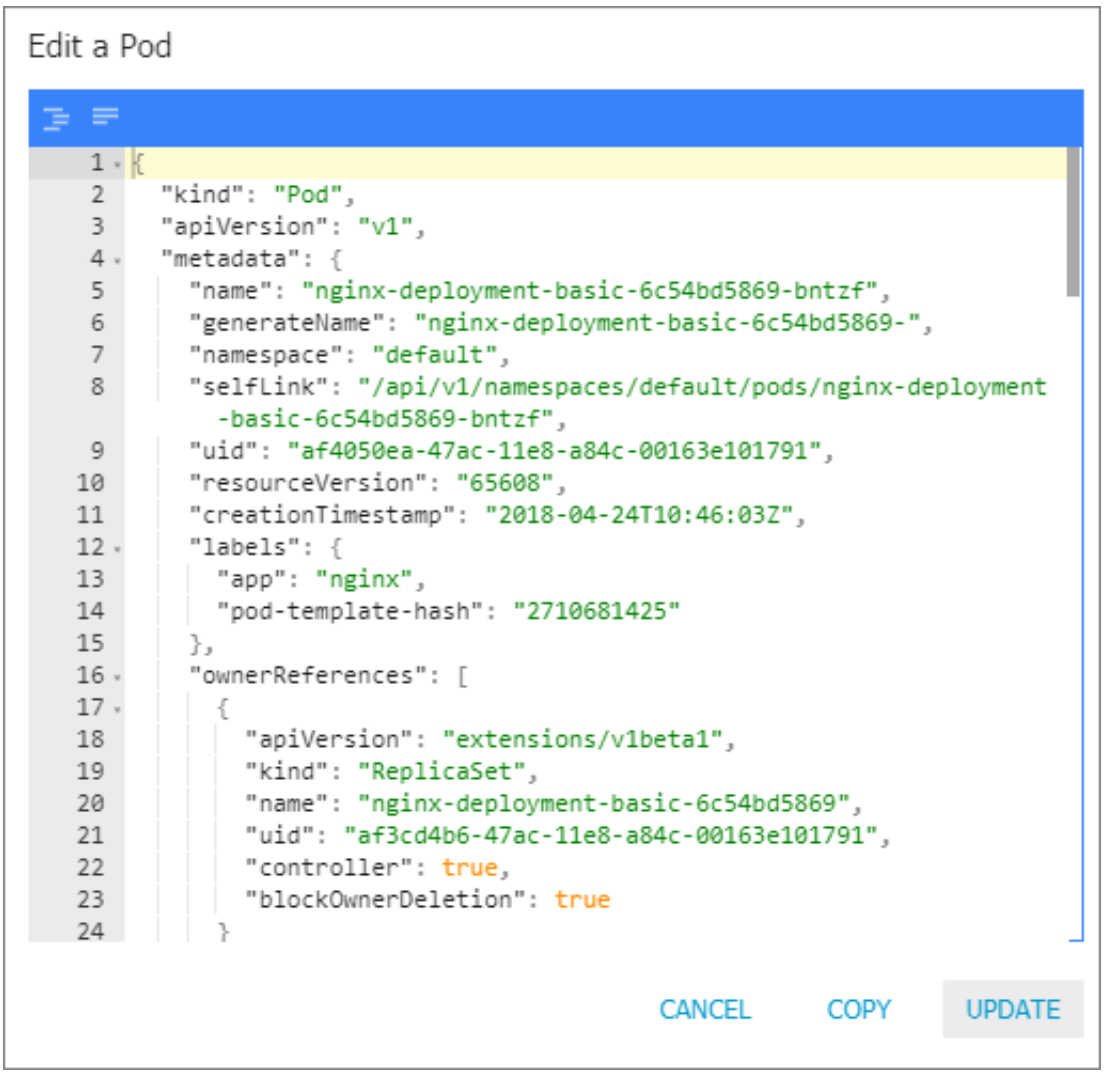
1. Log on to the *Container Service console*.
2. Click Kubernetes > **Clusters** in the left-side navigation pane.
3. Click **Dashboard** at the right of the cluster to enter the Kubernetes dashboard.



4. In the Kubernetes dashboard, click **Pods** in the left-side navigation pane.
5. Click the icon at the right of the pod and then select **View/edit YAML**.



6. The Edit a Pod dialog box appears. Change the container configurations and then click **UPDATE**.



1.5 Namespaces

1.5.1 Create a namespace

Prerequisites

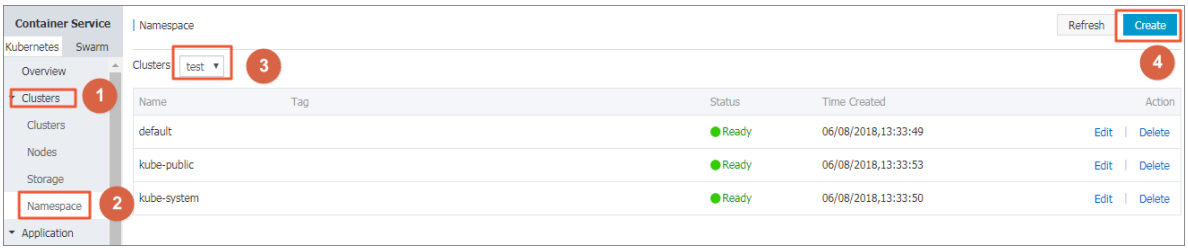
You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.

Context

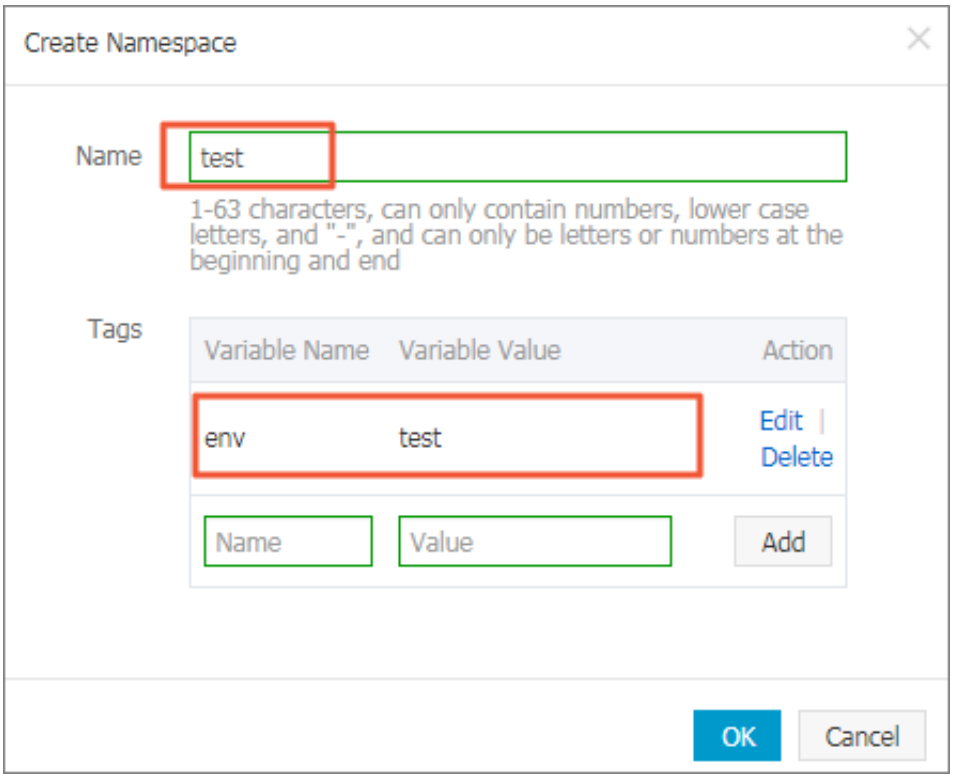
In Kubernetes clusters, you can use the Namespace function to create multiple virtual spaces. When the number of users in one cluster is large, multiple namespaces are used to divide the workspaces effectively and the cluster resources into different purposes. The namespace resources are assigned by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

Procedure

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



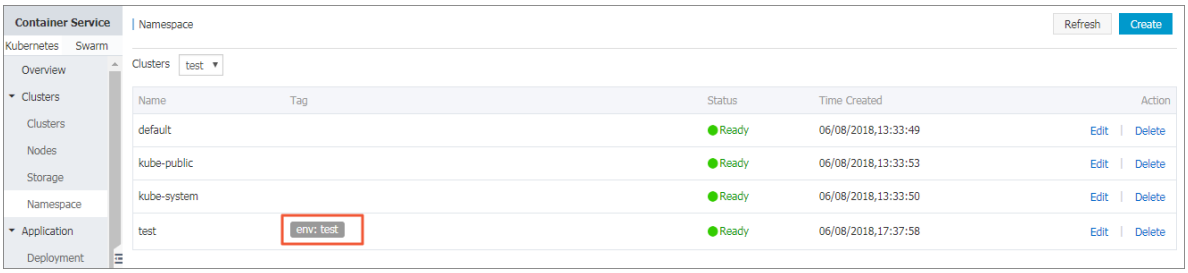
4. Configure the namespace in the displayed dialog box.



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

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

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



1.5.2 Configure resource quotas for namespaces

Prerequisites

- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a namespace **test**. For more information, see *Create a namespace*.
- Connect to the master node SSH IP address of the cluster. For more information, see *Access Kubernetes clusters by using SSH*.

Context

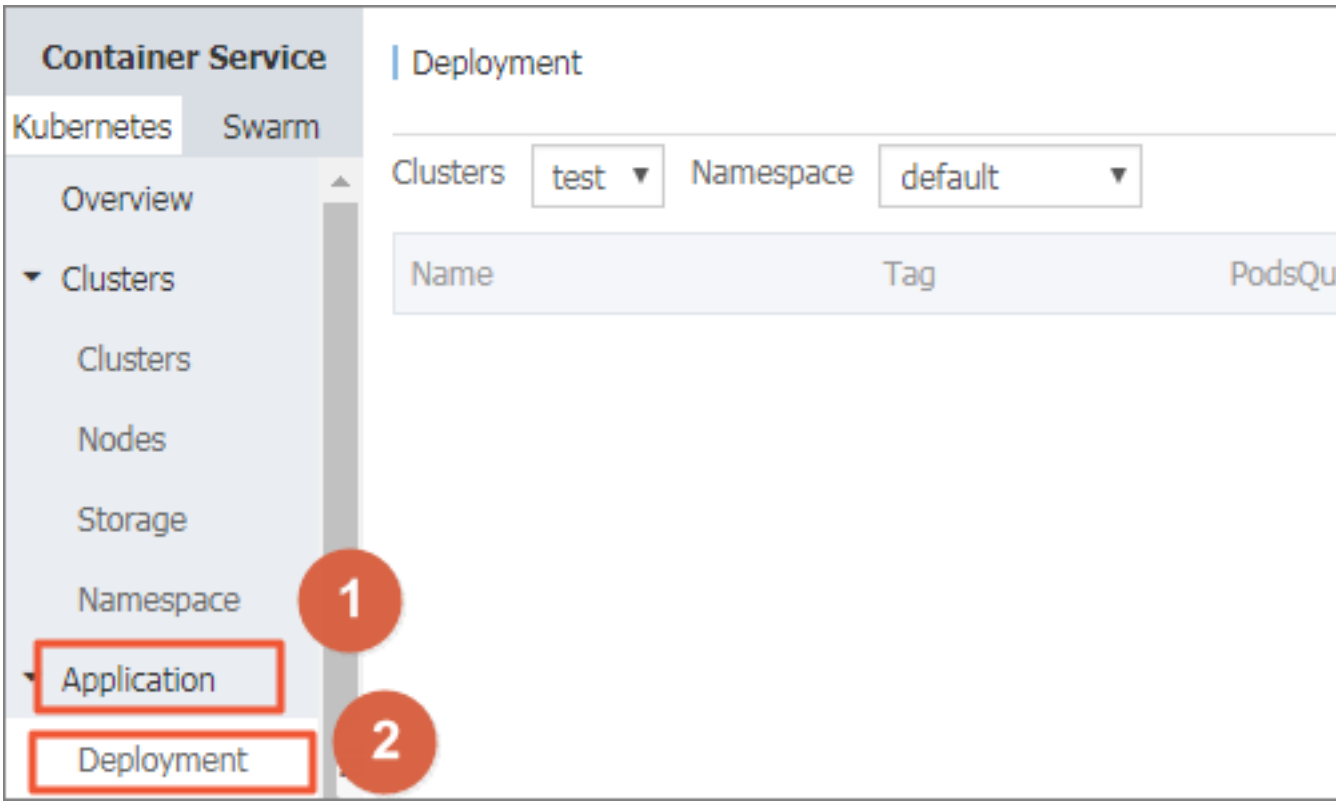
By default, a running pod can use the CPU and memory of nodes unlimitedly, which means any pod can use the computing resources of the cluster unlimitedly, and the pods of a namespace may use up the cluster resources.

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

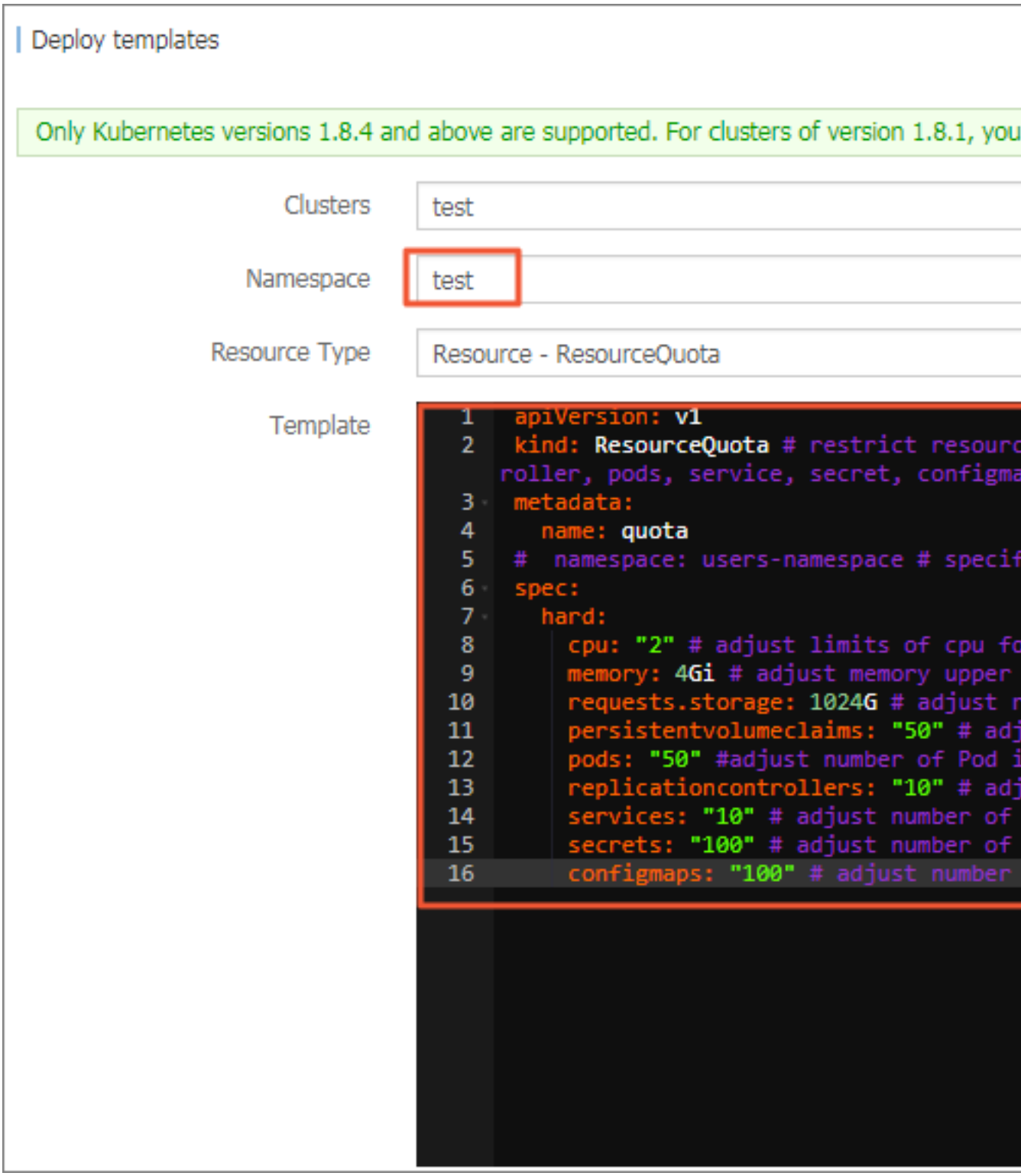
You can configure the resource quotas for a namespace, including CPU, memory, and number of pods. For more information, see *Resource Quotas*.

Procedure

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



3. On the Deploy templates page, select the cluster and namespace (**test** in this example) from the Clusters and Namespace drop-down lists. Use a custom template or the example template Resource – ResourceQuota.



You must configure the ResourceQuota template according to your cluster resources and the plan for the namespace resources. In this example, the template is as follows:

```
apiVersion: v1
 kind: ResourceQuota # restrict resource quota for cpu, memory
, storage, pvc, replicationcontroller, pods, service, secret,
configmap
```

```
 metadata:
   name: quota
 # namespace: users-namespace # specify your namespace to apply
resource quota
 spec:
   hard:
     cpu: "2" # adjust limits of cpu for your namespace
     memory: 4Gi # adjust memory upper limits for your namesapce
     requests.storage: 1024G # adjust request of storage size for
your namespace
     persistentvolumeclaims: "50" # adjust number of pvc for your
namespace
     pods: "50" #adjust number of Pod in your namespace
     replicationcontrollers: "10" # adjust number of Replicatio
nController in your namespace
     services: "10" # adjust number of service for your namespace
     secrets: "100" # adjust number of secrets for your namespace
     configmaps: "100" # adjust number of configmap for your
namespace
```

4. You have configured the resource quotas for this namespace. Connect to the master node SSH IP address and run the following command to view the resource quotas and usage of this namespace.

```
# kubectl describe quota quota --namespace=test
 Name: quota
 Namespace: test
 Resource Used Hard
 --------                 ----  ----
 configmaps 0 100
 cpu 0 2
 memory 0 4Gi
 persistentvolumeclaims 0 50
 pods 0 50
 replicationcontrollers 0 10
 requests.storage 0 1024G
 secrets 1 100
 services 0 10
```

1.5.3 Update a namespace

Prerequisites

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

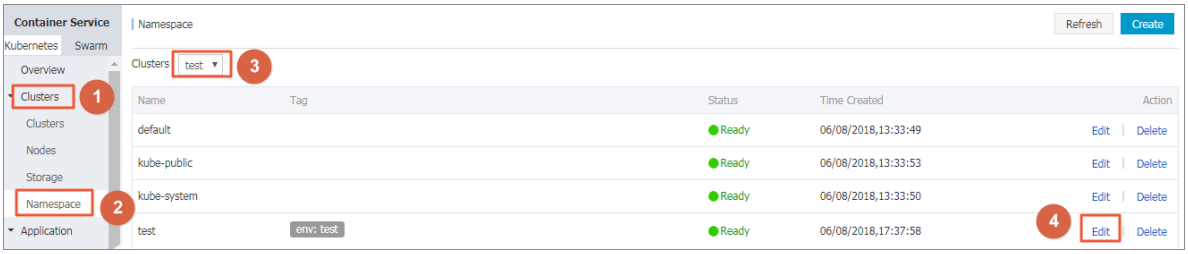
Context

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

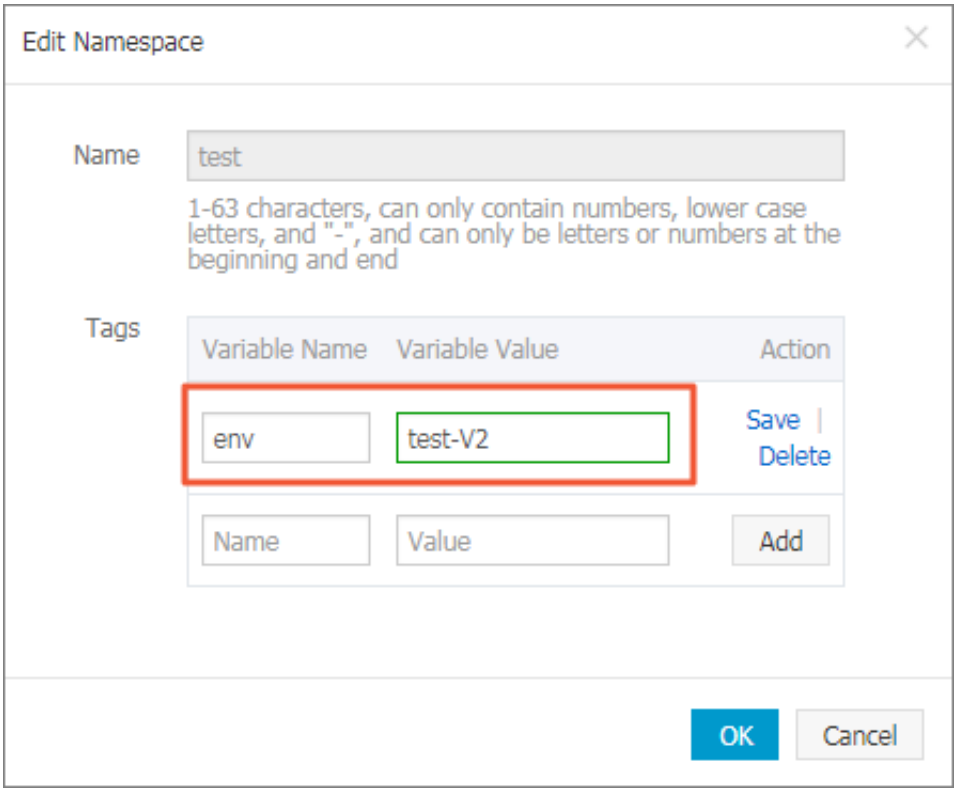
Procedure

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

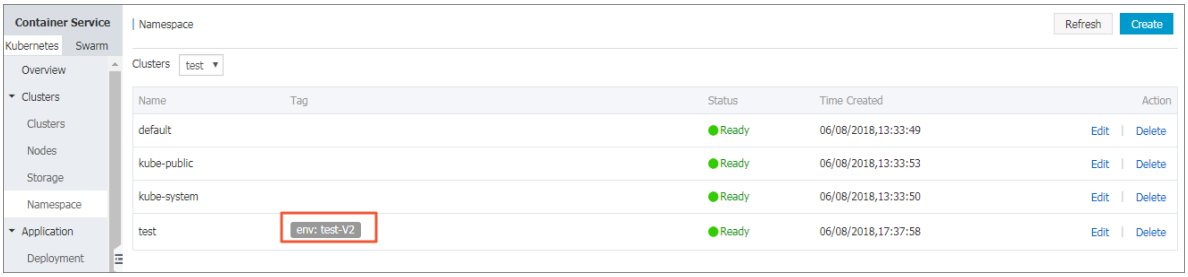
2. Under Kubernetes, click**Clusters** > **Namespace**in the left-side navigation pane.
3. Select the cluster from the Clusters drop-down list and click **Edit** at the right of the cluster.



4. Update the namespace tags in the displayed dialog box. For example, change the tag to `env: test-V2`.



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



1.5.4 Delete a namespace

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

Prerequisites

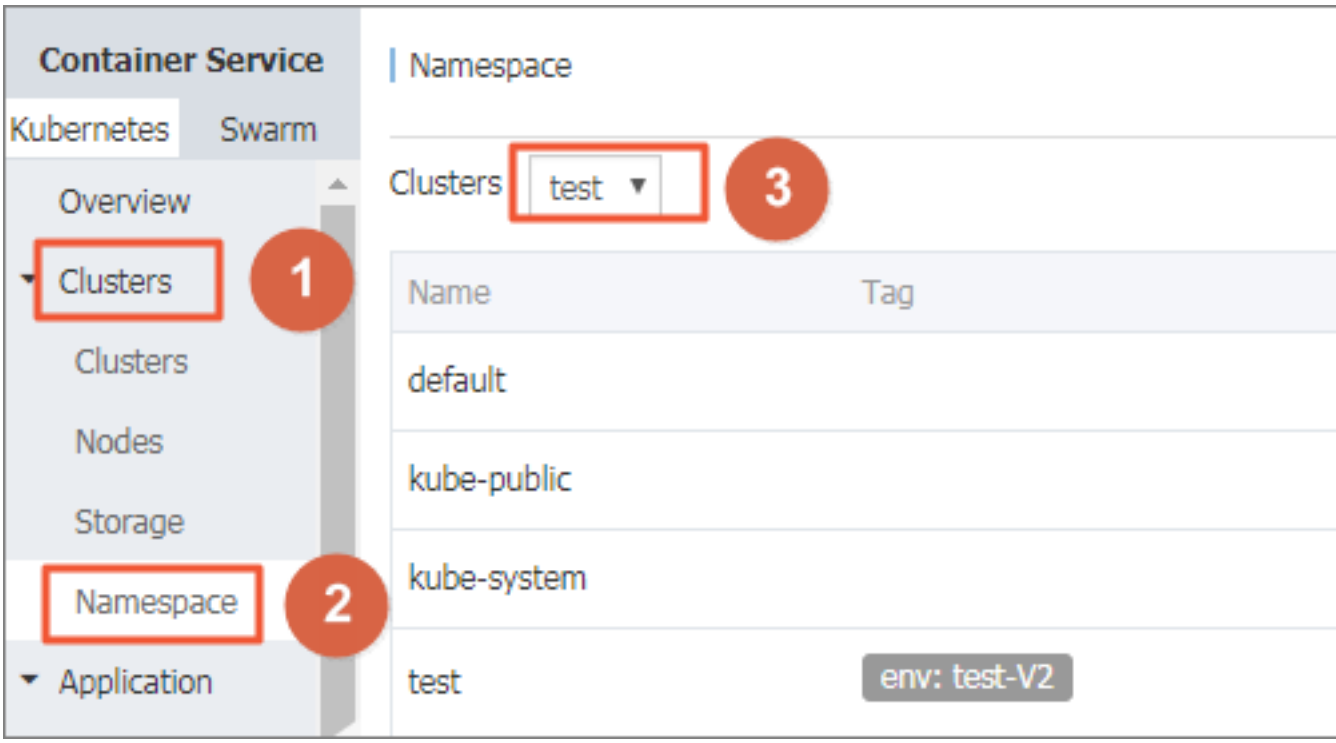
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a namespace **test**. For more information, see *Create a namespace*.

Context

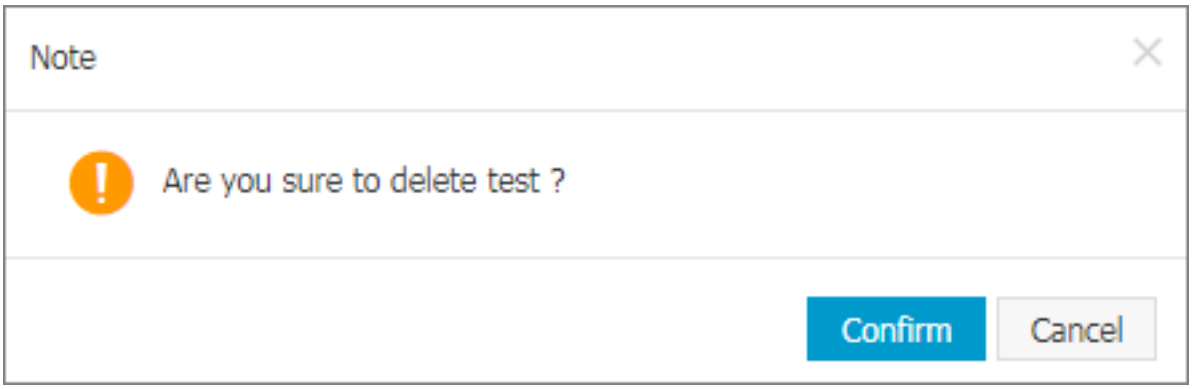
Note:
Deleting a namespace also deletes all of its resource objects, so proceed with caution.

Procedure

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



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



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

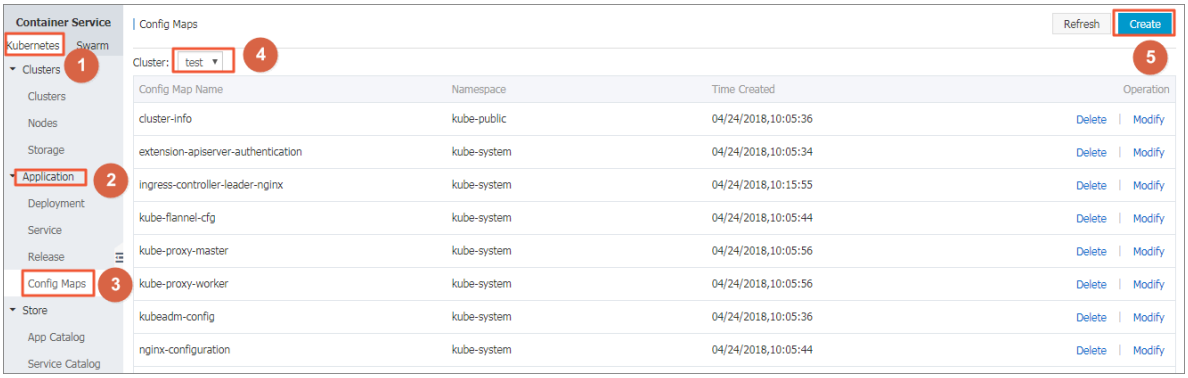
1.6 Config map

1.6.1 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.

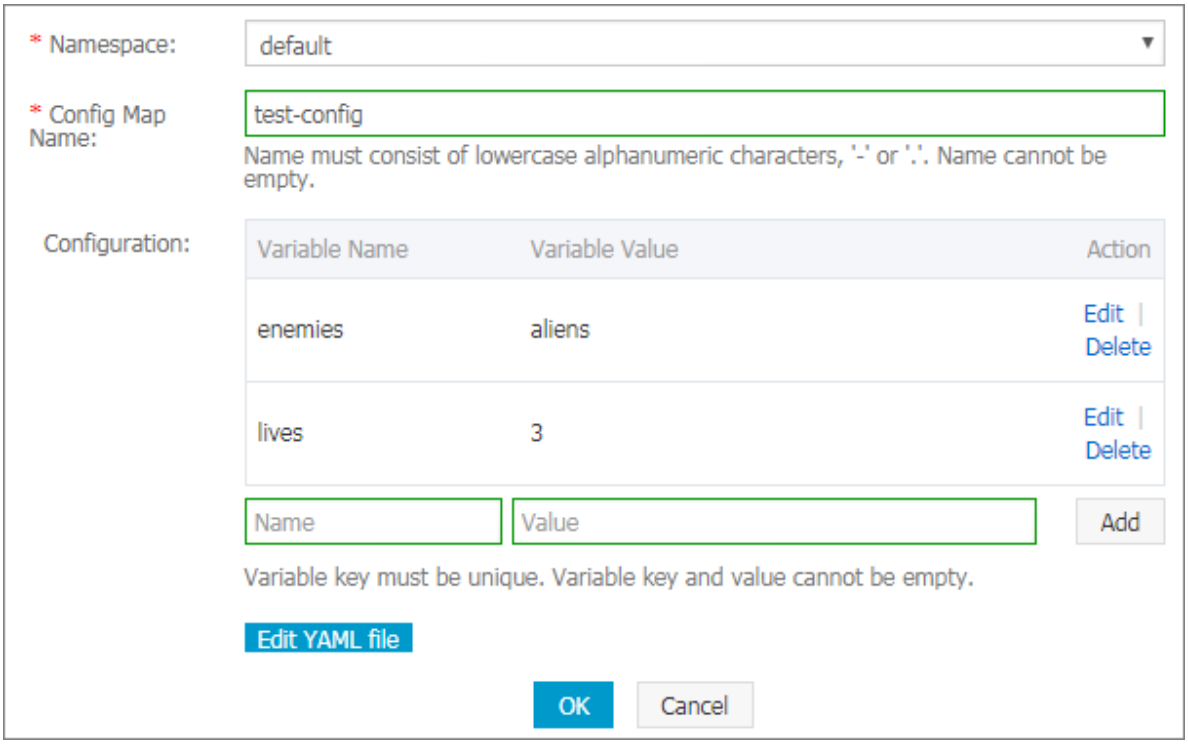
Create a config map on Config Maps page

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.



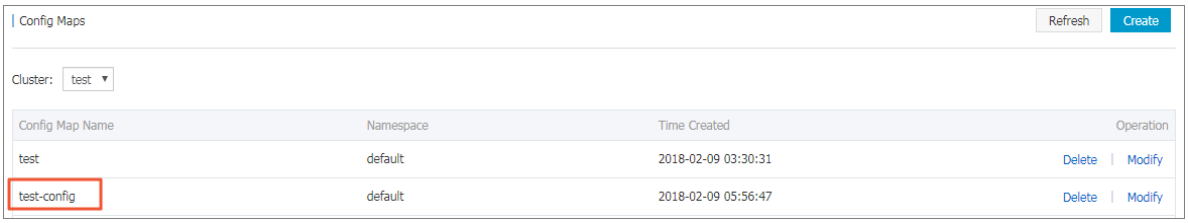
4. Complete the settings and then click **OK**.
   - **Namespace**: Select the namespace to which the config map belongs. The config map is a Kubernetes resource object and must act on a namespace.
   - **Config Map Name:** Enter the config map name, which can contain lowercase letters, numbers, hyphens (-), 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 YAML file** to set the configurations in the displayed dialog box, and then click **OK**.



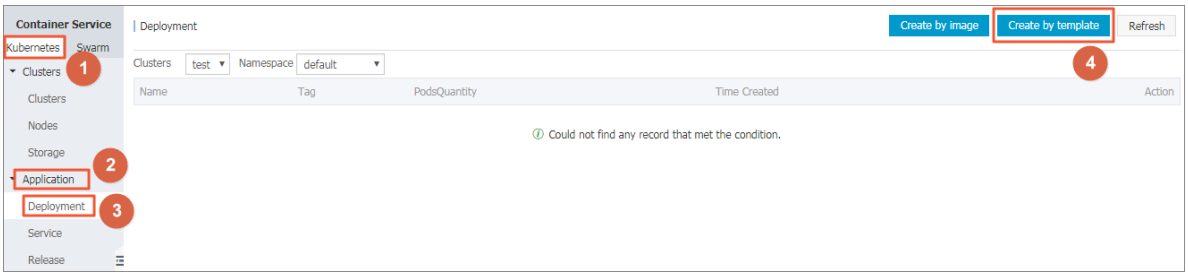
In this example, configure the variables enemies and lives to pass the parameters aliens and 3 respectively.

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



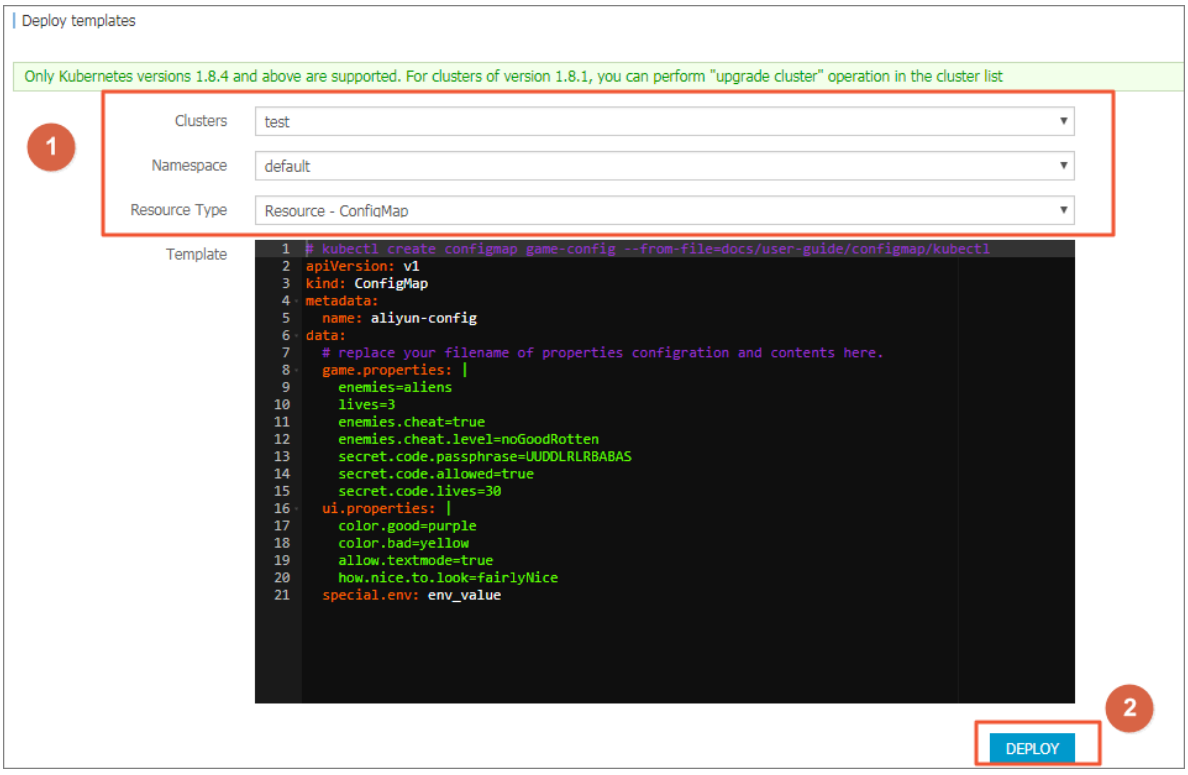
Create a config map by using a template

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.

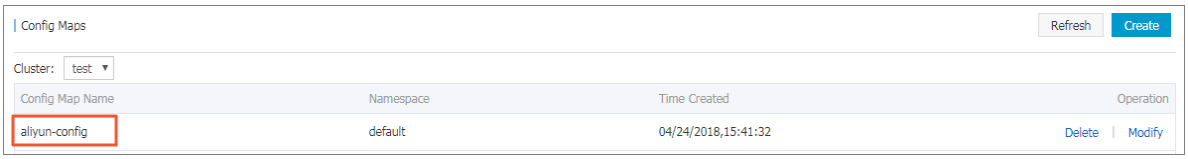


4. On the Deploy templates page, complete the settings and then click **DEPLOY**.

   - **Clusters:** Select the cluster in which the config map is to be created.
   - **Namespace:** Select the namespace to which the config map belongs. The config map is a Kubernetes resource object and must act on a 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.



1.6.2 Use a config map in a pod

You can use a config map in a pod in the following scenarios:

- Use a config map to define the pod environment variables.
- Use a config map to configure command line parameters.
- Use a config map in data volumes.

For more information, see *Configure a pod to use a ConfigMap*.

Limits

To use a config map in a pod, make sure the config map and the pod are in the same cluster and namespace.

Create a config map

In this example, create a config map special-config, which includes two key-value pairs: `SPECIAL_LEVEL: very` and `SPECIAL_TYPE: charm`.

Create a config map by using an orchestration template

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

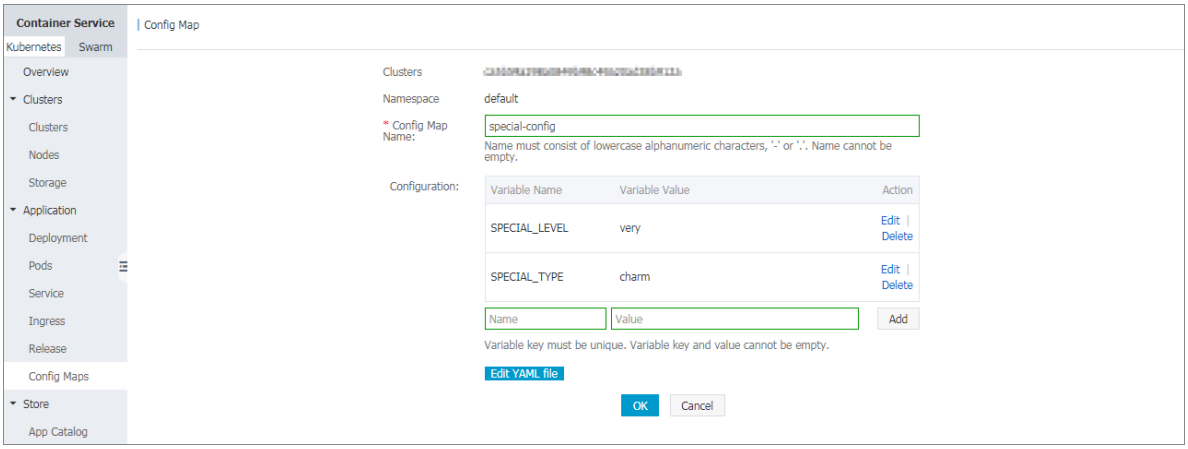
   You can use the following YAML sample template to create a config map.

```
apiVersion: v1
 kind: ConfigMap
 metadata:
   name: special-config
   namespace: default
 data:
   SPECIAL_LEVEL: very
   SPECIAL_TYPE: charm
```

Create a config map on Config Maps page

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

2. Under Kubernetes, click **Application >** > **Configuration item**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, click **Application >** > **Deployment**Click **Create by template** in the upper-right corner.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Select a sample template or Custom from the Resource Type drop-down list. Click **DEPLOY**.

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

   See the following orchestration example:

```
apiVersion: v1
 kind: Pod
 metadata:
   name: config-pod-1
 spec:
   containers:
     - name: test-container
       image: busybox
       command: [ "/bin/sh", "-c", "env" ]
       env:
         - name: SPECIAL_LEVEL_KEY
           valueFrom: ##Use valueFrom to specify env to reference
the value of the config map.
             configMapKeyRef:
```

```
                name: special-config ##The referenced config map name
.
                key: SPECIAL_LEVEL ##The referenced config map key.
   restartPolicy: Never
```

Similarly, to define the values of multiple config maps to the environment variable values of the pod, add multiple env parameters in the pod.

Configure all key-value pairs of a config map to pod environment variables

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

   To configure all the key-value pairs of a config map to the environment variables of a pod, use the envFrom parameter. The key in a config map becomes the environment variable name in the pod.

   See the following orchestration example:

```
apiVersion: v1
 kind: Pod
 metadata:
   name: config-pod-2
 spec:
   containers:
     - name: test-container
       image: busybox
       command: [ "/bin/sh", "-c", "env" ]
       envFrom: ## Reference all the key-value pairs in the config
map special-config.
       - configMapRef:
           name: special-config
   restartPolicy: Never
```

Use a config map to configure command line parameters

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

   You can use the config map to configure the commands or parameter values in the container by using the environment variable replacement syntax `$(VAR_NAME)`.

See the following orchestration example:

```
apiVersion: v1
 kind: Pod
 metadata:
   name: config-pod-3
 spec:
   containers:
     - name: test-container
       image: busybox
       command: [ "/bin/sh", "-c", "echo $(SPECIAL_LEVEL_KEY) $(
SPECIAL_TYPE_KEY)" ]
       env:
         - name: SPECIAL_LEVEL_KEY
           valueFrom:
             configMapKeyRef:
               name: special-config
               key: SPECIAL_LEVEL
         - name: SPECIAL_TYPE_KEY
           valueFrom:
             configMapKeyRef:
               name: special-config
               key: SPECIAL_TYPE
   restartPolicy: Never
```

The output after running the pod is as follows:

```
very charm
```

Use a config map in data volumes

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

   You can also use a config map in data volumes. Specifying the config map name under volumes stores the key-value pair data to the mountPath directory (*/etc/config* in this example). Then, the configuration file with key as the name and value as the contents is generated.

   See the following orchestration example:

```
apiVersion: v1
 kind: Pod
 metadata:
   name: config-pod-4
 spec:
   containers:
     - name: test-container
       image: busybox
```

```
        command: [ "/bin/sh", "-c", "ls /etc/config/" ] ##List the
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
```

1.6.3 Update a config map

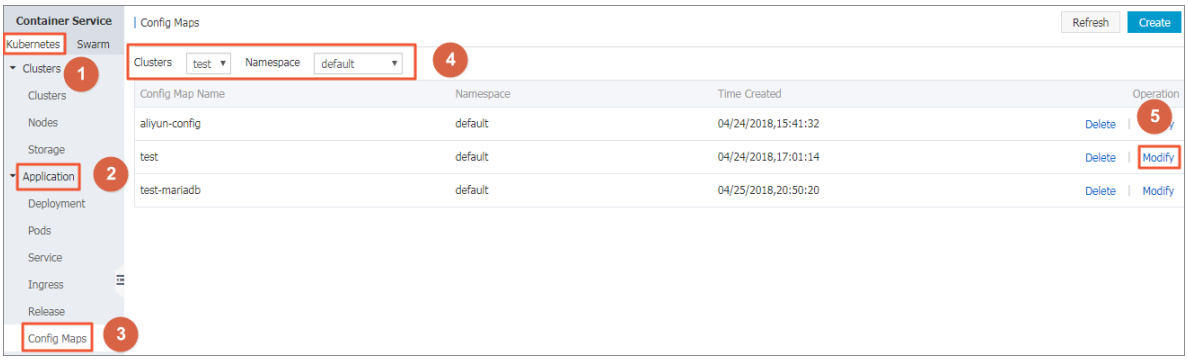
You can modify the configurations of a config map.

Note:

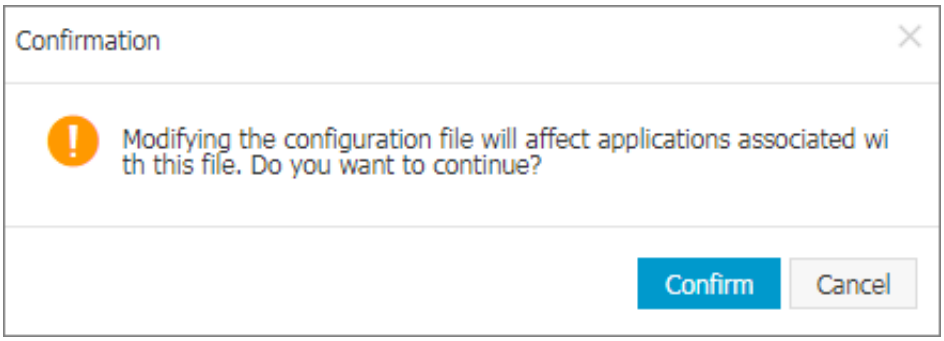
Updating a config map affects applications that use this config map.

Update a config map on Config Maps page

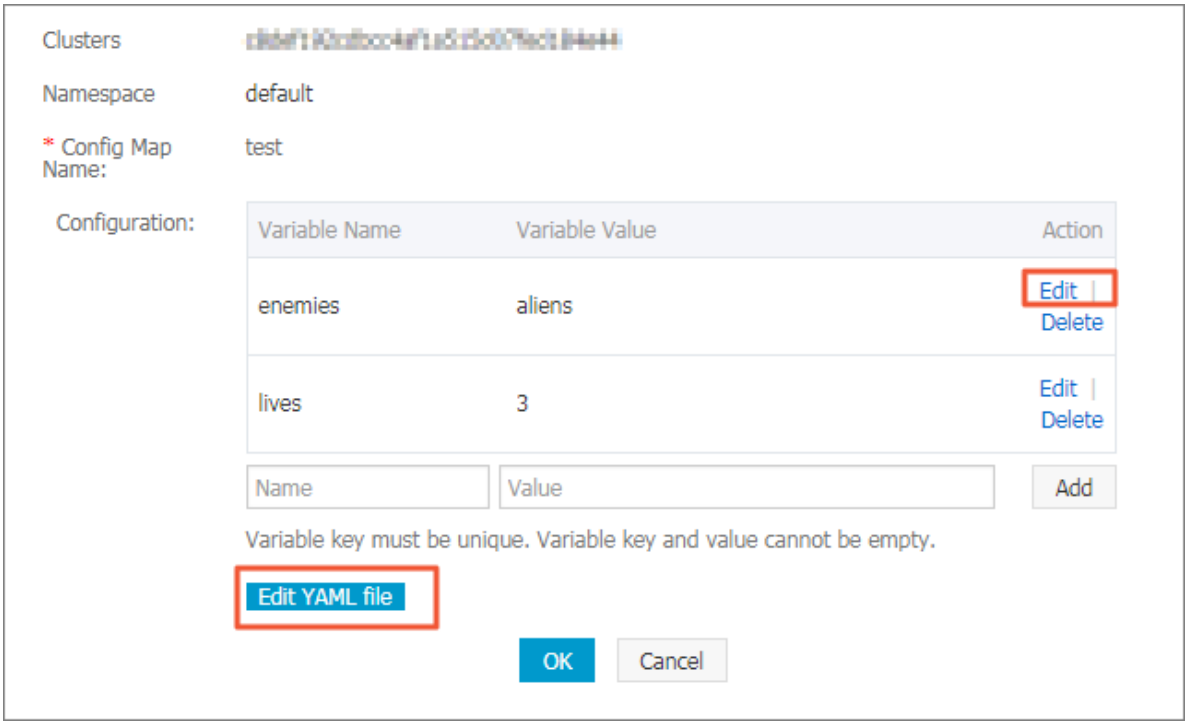
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application >** > **Config Maps** in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Modify** at the right of the config map.



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



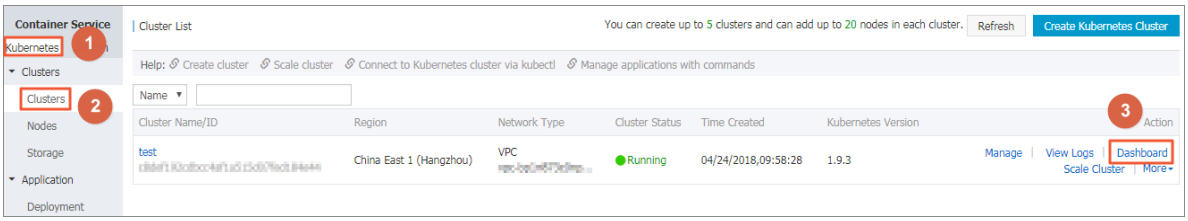
5. Modify the configurations.
   - Click **Edit** on the right of the configuration you want to modify. Update the configuration and then click **Save**.
   - You can also click **Edit YAML file**. Click **OK** after making the modifications.



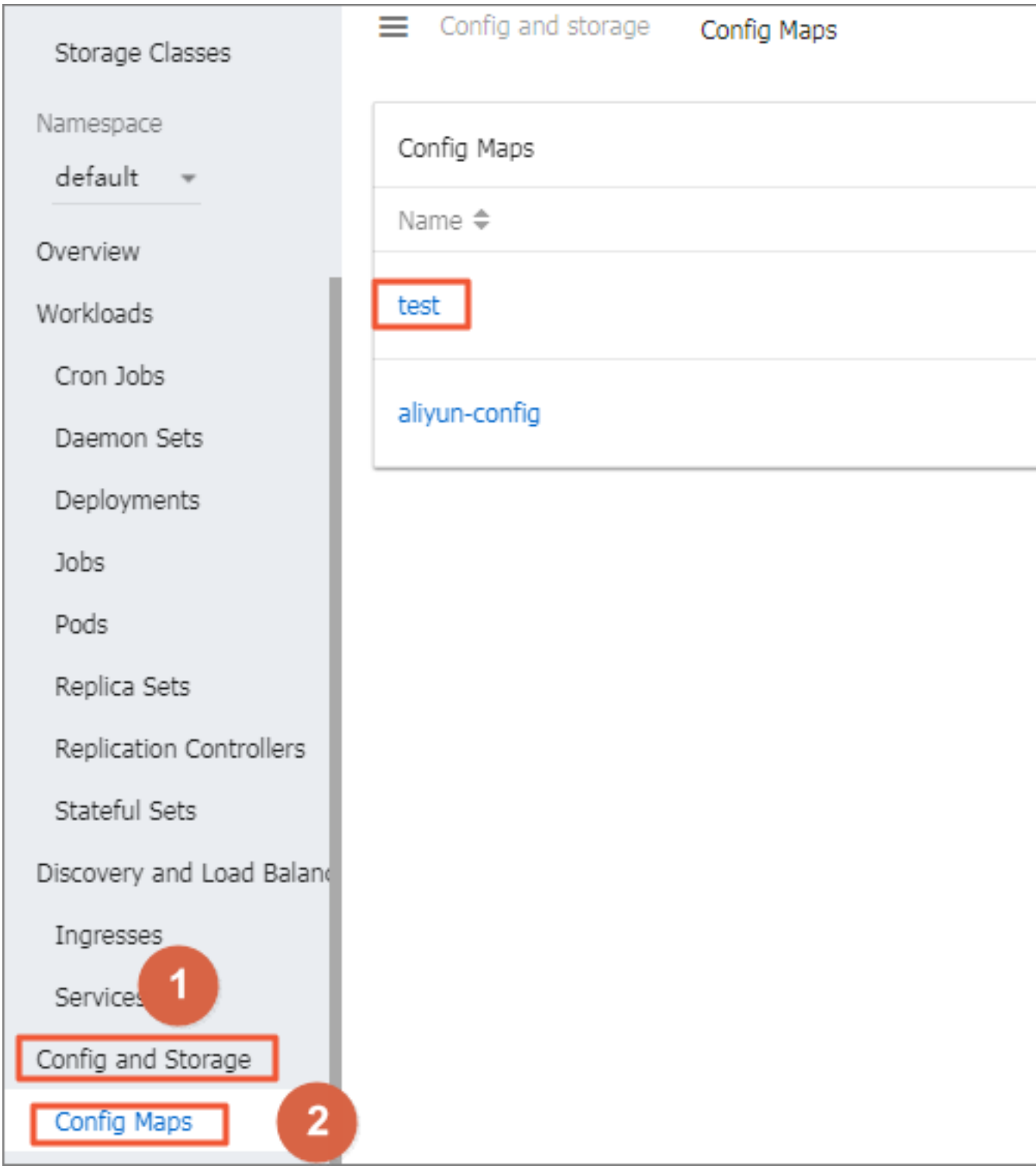
6. After modifying the configurations, click **OK**.

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 **Dashboard**at the right of the cluster.



4. In the Kubernetes dashboard, click**Config and Storage >** > **Config Maps in the left-side navigation pane.**Click the icon at the right of the config map and then**select** > **View/edit YAML** .



5. The Edit a Config Map dialog box appears. Modify the configurations and then click **UPDATE**.

1.7 Secrets

1.7.1 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:

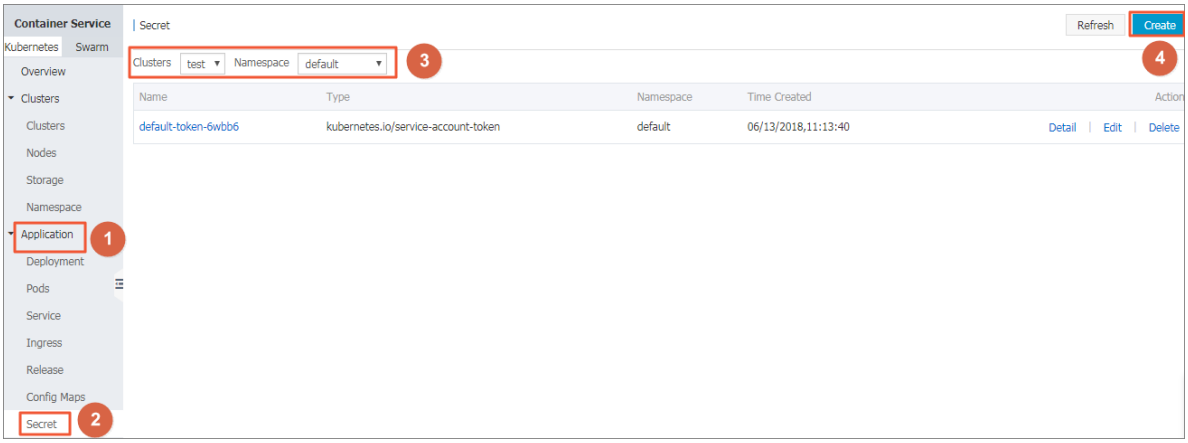
- Service Account: Automatically created by Kubernetes, which is used to access Kubernetes APIs and is automatically mounted to the pod directory `/run/secrets/kubernetes.io/serviceaccount`.
- Opaque: Secret in the base64 encoding format, which is used to store sensitive information such as passwords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

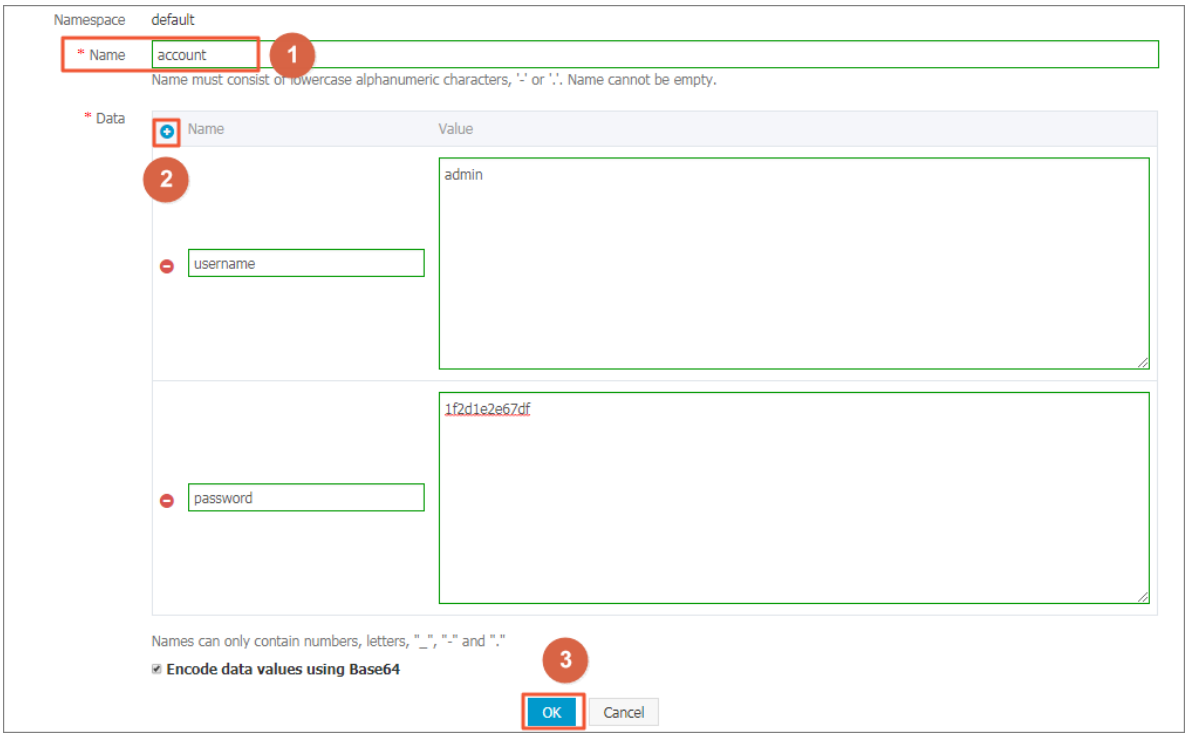
1. Log on to the *Container Service console*.
2. Under Kubernetes, click**Application** > **Secret**in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Create** in the upper-right corner.



4. Complete the configurations to create a secret.

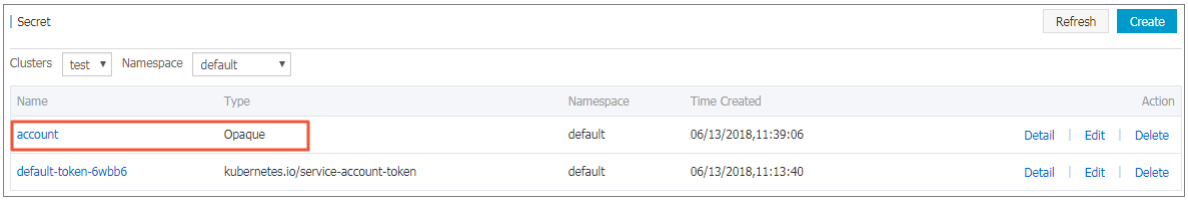
Note:

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



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

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



1.7.2 View secret details

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

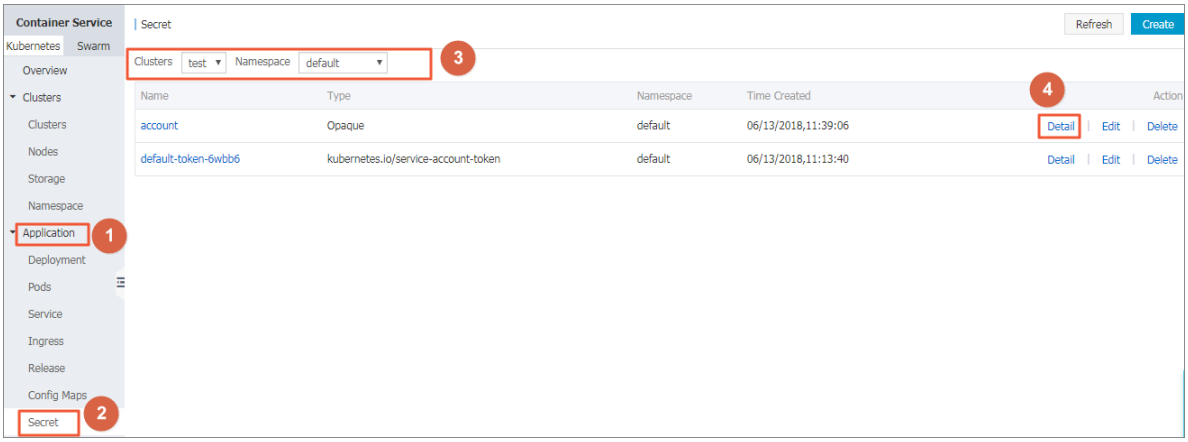
Prerequisites

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

Context

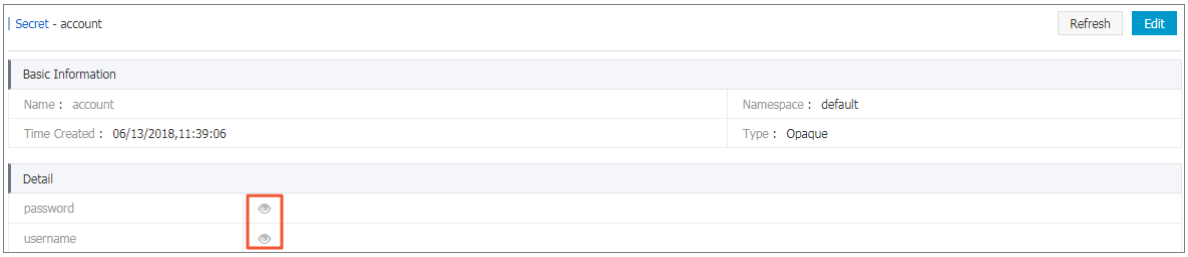
Procedure

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



1.7.3 Update a secret

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

Prerequisites

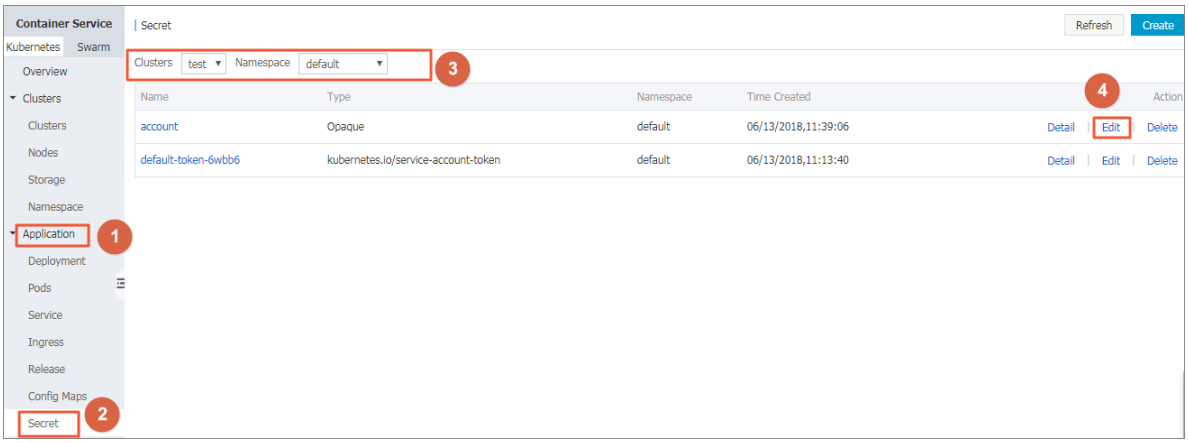
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a secret. For more information, see *Create a secret*.

Context

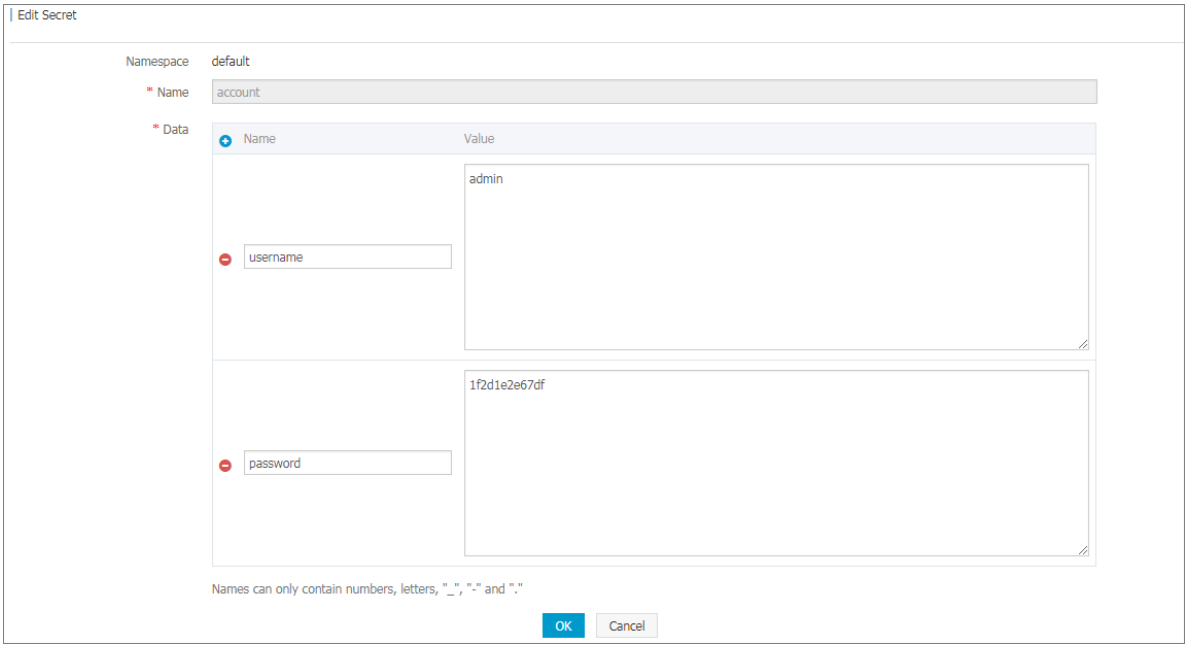
Procedure

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

2. Under Kubernetes, click**Application** > **Secret**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**.

1.7.4 Delete a secret

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

Prerequisites

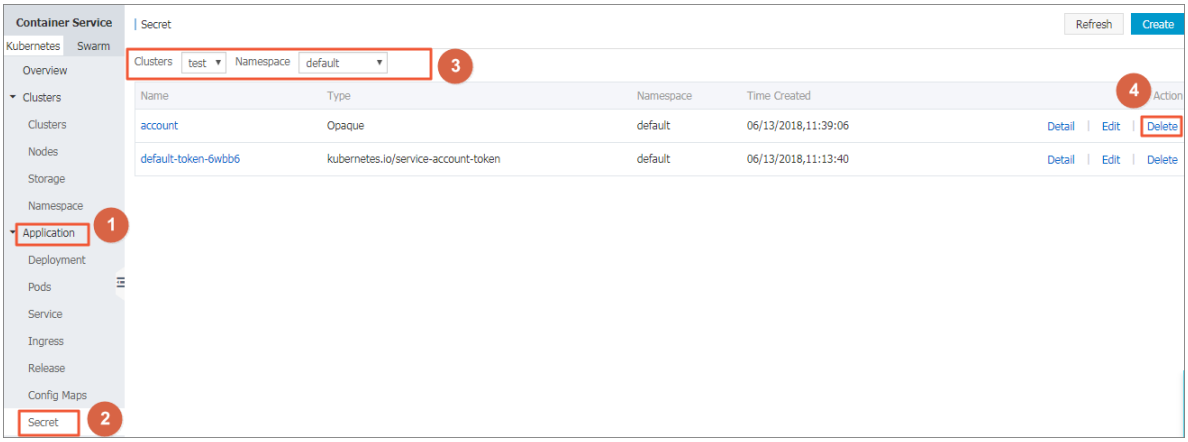
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- You have created a secret. For more information, see *Create a secret*.

Context

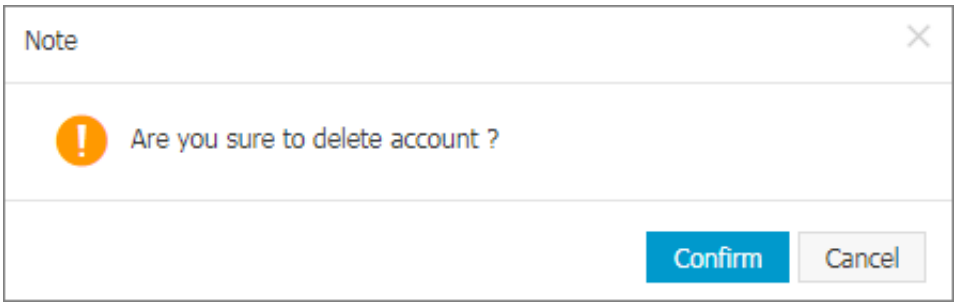
> **Note:**
> Do not delete the secret generated when the cluster is created.

Procedure

1. Log on to the *Container Service console*.
2. Under Kubernetes, click**Application** > **Secret**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 to delete the secret.



1.8 Ingress

1.8.1 Ingress configurations

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

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

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

Annotation

You can configure an ingress annotation, specifying the ingress controller to use, rules for routing, such as routing weight rules, grayscale publish, and rewrite rules. For more information, see *https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/*.

For example, a typical rewrite annotation `nginx.ingress.kubernetes.io/rewrite-target: /` redirects the path */path* to the path `/` that can be recognized by the backend services.

Rules

The rules indicate those that authorize the inbound access to the cluster and are generally the HTTP rules, including the domain name (virtual hostname), URL access path, service name, and port.

You must complete the following configurations for each HTTP rule:

- Host: Enter the testing domain name of an Alibaba Cloud Kubernetes cluster or a virtual hostname, such as `foo.bar.com`.
- Path: Specify the URL path of the service access. Each path is associated with a backend service. Before Alibaba Cloud Server Load Balancer forwards the traffic to the backend, all inbound requests must match with the domain name and path.
- Backend configuration: Service configuration that is a combination of `service:port` and traffic weight. The Ingress traffic is forwarded to the matched backend services based on the traffic weight.
  - Name: The name of the backend service forwarded by Ingress.
  - Port: The port exposed by the service.
  - Weight: The weight rate of each service in a service group.

> **Note:**
>
> 1. The service weight is calculated in relative values. For example, if both service weights are set to 50, the weight ratio of both services is 50%.
> 2. A service group (a service with the same Host and Path in the same ingress yaml) has a default weight value of 100 and the weight is not explicitly set.

Grayscale publish

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

> **Note:**
>
> Currently, the Alibaba Cloud Container Service Kubernetes Ingress Controller requires **`0.12.0-5`** and above to support the traffic segmentation feature.

1. Traffic segmentation based on the request header.
2. Traffic segmentation based on cookie.
3. Traffic segmentation based on query (request) parameters.

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

TLS

You can encrypt the Ingress by specifying a secret that contains the TLS private key and certificate to implement the secure Ingress access. The TLS secret must contain the certificate named tls.crt and private key named tls.key. For more information about the TLS principles, see *TLS*. For how to create a secret, see *Configure a safe routing service*.

Tag

You can add tags for Ingress to indicate the characteristics of the Ingress.

1.8.2 Create an Ingress in Container Service console

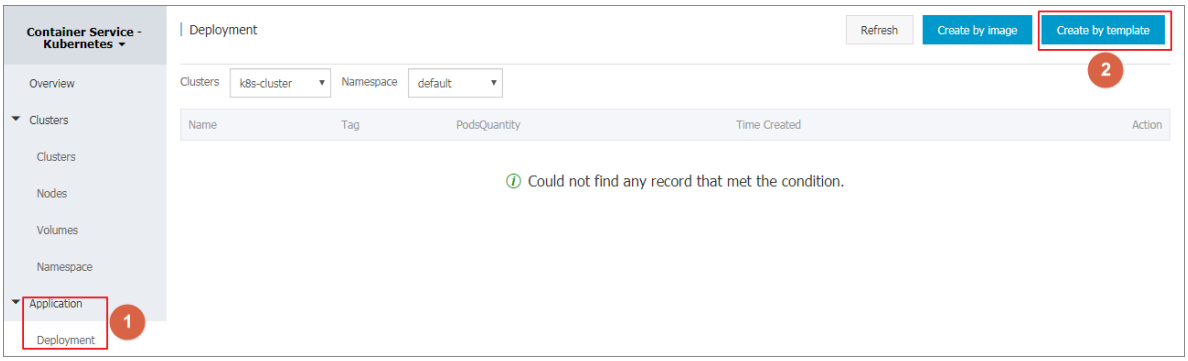
Alibaba Cloud Container Service console integrates with the Ingress service, which allows you to quickly create an Ingress service in the Container Service console to build the flexible and reliable traffic access layer.

Prerequisites

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

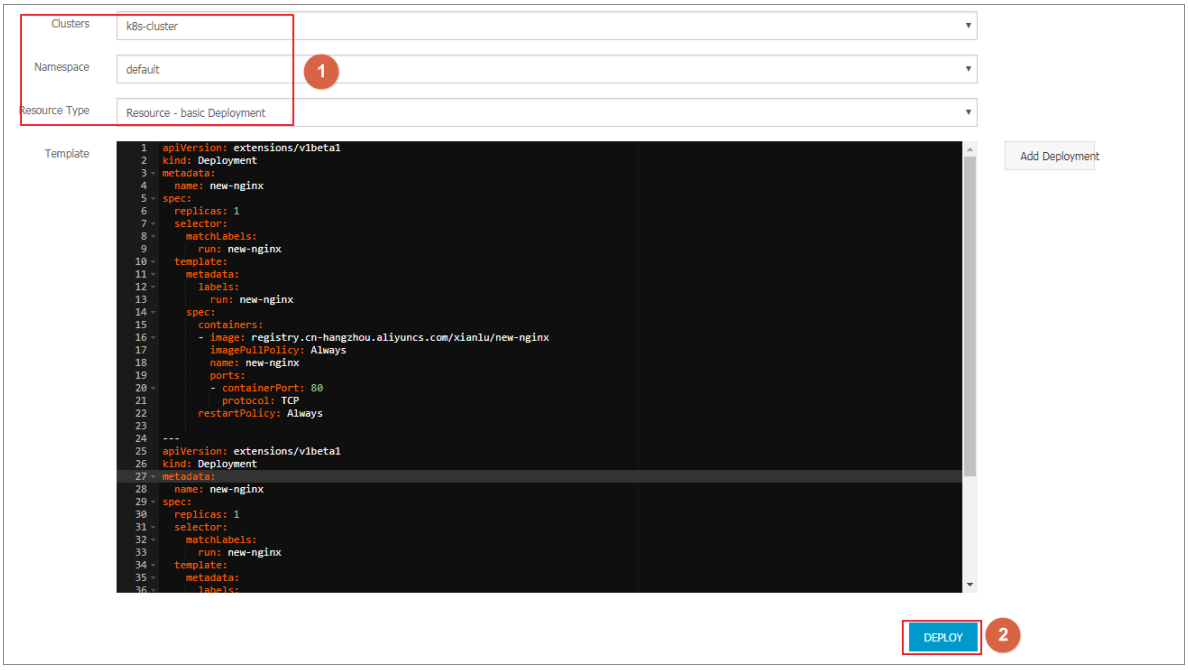
Step 1. Create a deployment and a service

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



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
```

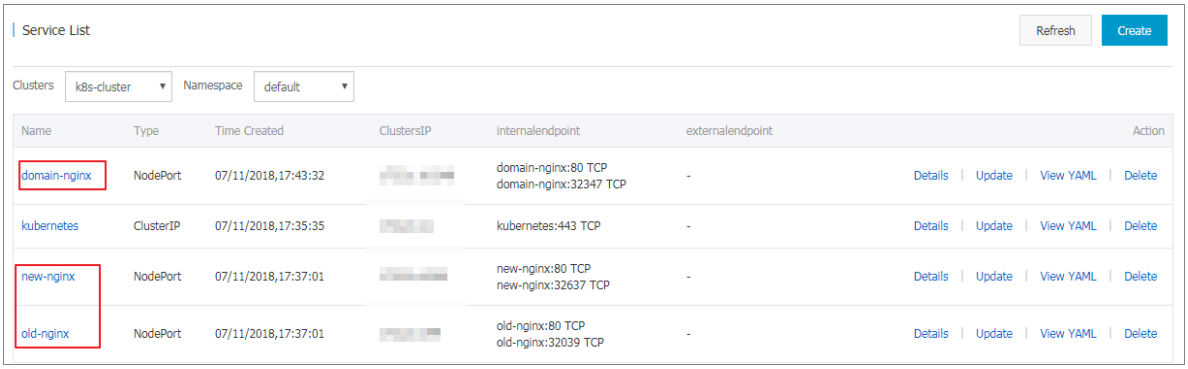
The orchestration template for domain-nginx is as follows:

```
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
```

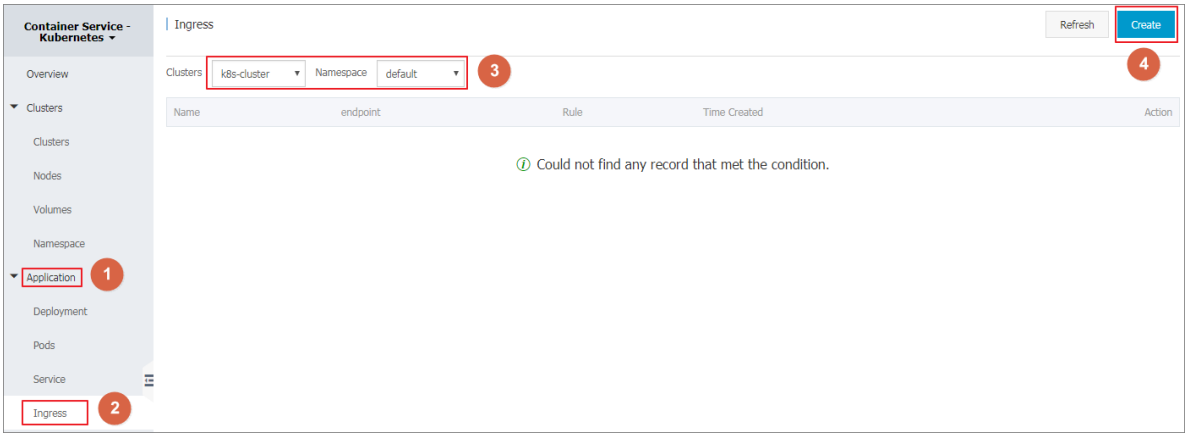
5. Click **Application** > **Service** in the left-side navigation pane to enter the Services List page.

   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.

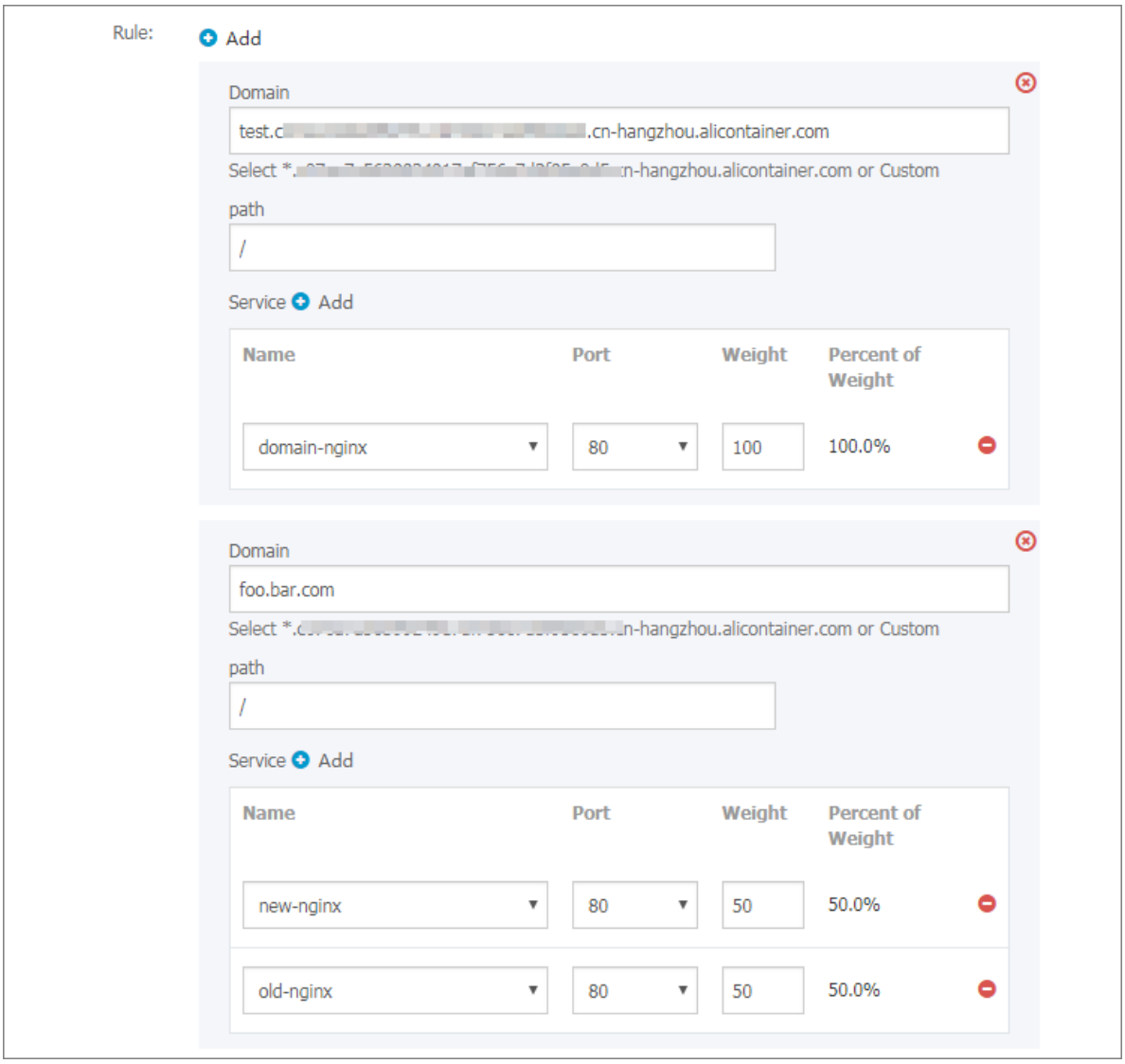


4. In the displayed dialog box, enter the Ingress name. In this example, enter nginx-ingress.

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.

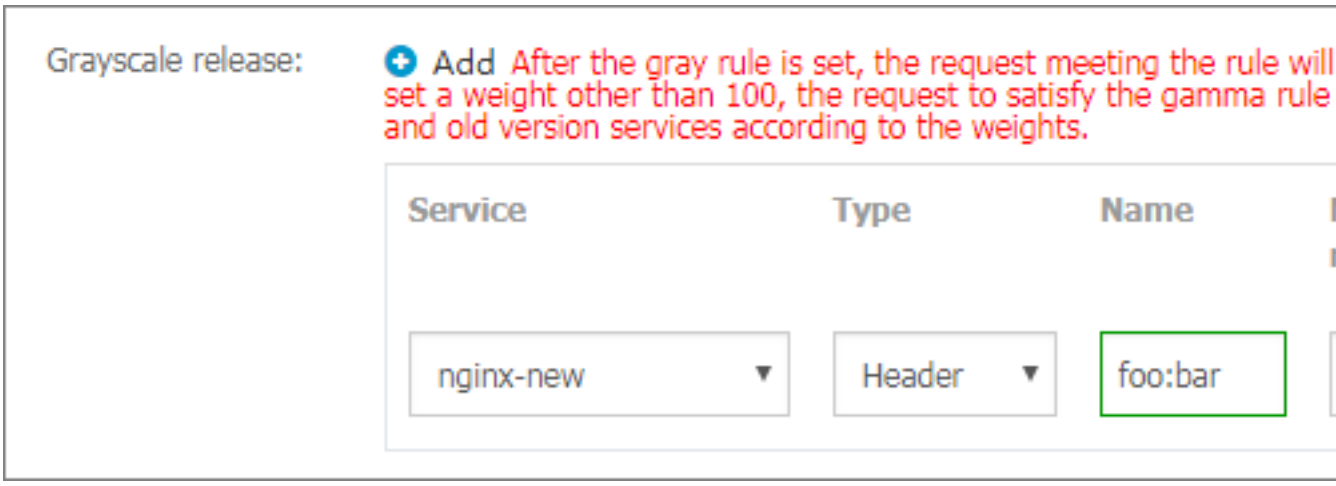
   Note:
   Currently, the Alibaba Cloud Container Service Kubernetes Ingress Controller requires **`0.12.0-5`** and above to support the traffic segmentation feature.

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

   1. Traffic segmentation based on the request header.
   2. Traffic segmentation based on cookie.
   3. Traffic segmentation based on query (request) parameters.

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



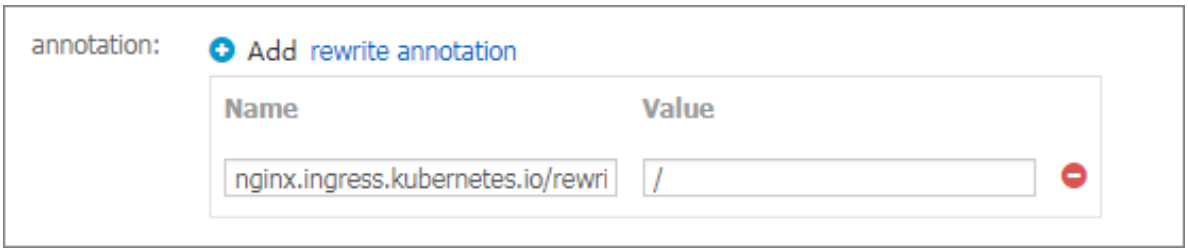
- **Service**: Routing rule configuration service.
- **Type**: matching request header, cookie, and query (request) parameters are supported.
- **Name and match value**: User-defined request field, name and match value are key-value pairs.
- **Match rules**: Regular and exact matches are supported.

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.

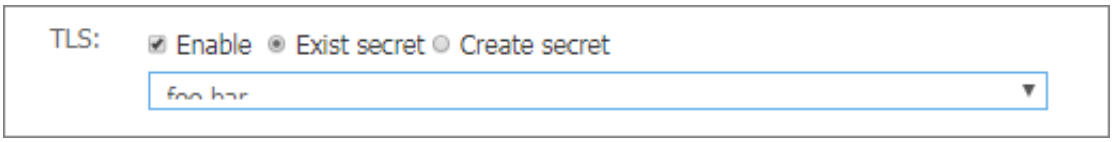
Note:
In this example, the access path is not configured, so no need to configure rewrite annotations. The purpose of the rewrite annotation is to enable Ingress to forward to the backend as the root path, avoiding 404 errors caused by incorrect access path configuration.

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.



     1. Log on to the master node and create *tls.key* and *tls.crt*.

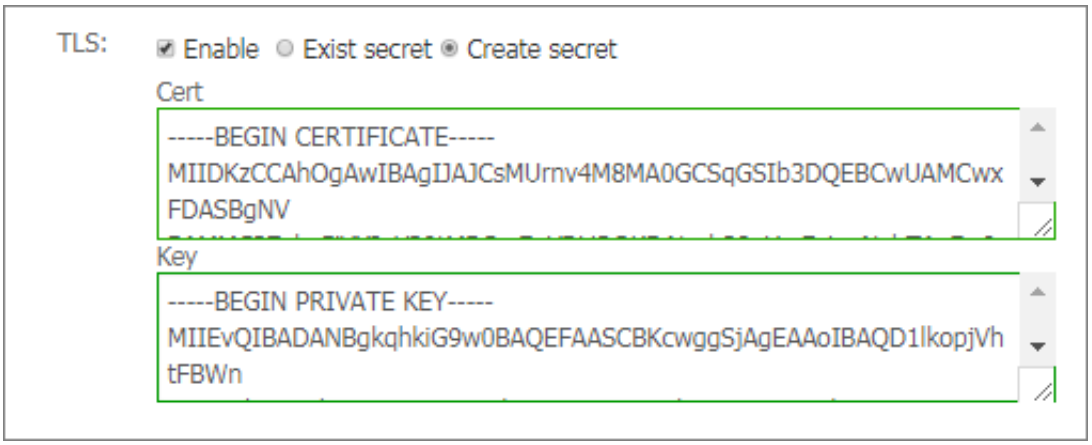
```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls
.key -out tls.crt -subj "/CN=foo.bar.com/O=foo.bar.com"
```

     2. Create a secret.

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

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

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



1. Log on to the master node and create *tls.key* and *tls.crt*.

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

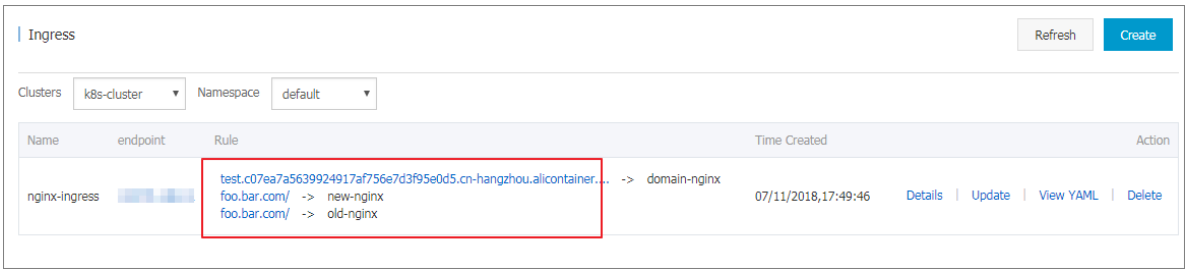
2. Run the `vim tls.key` and `vim tls.crt` to get the generated private key and certificate.
3. Copy the generated certificate and private key to the Cert and Key fields.

9. Adding the tags.

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

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
```

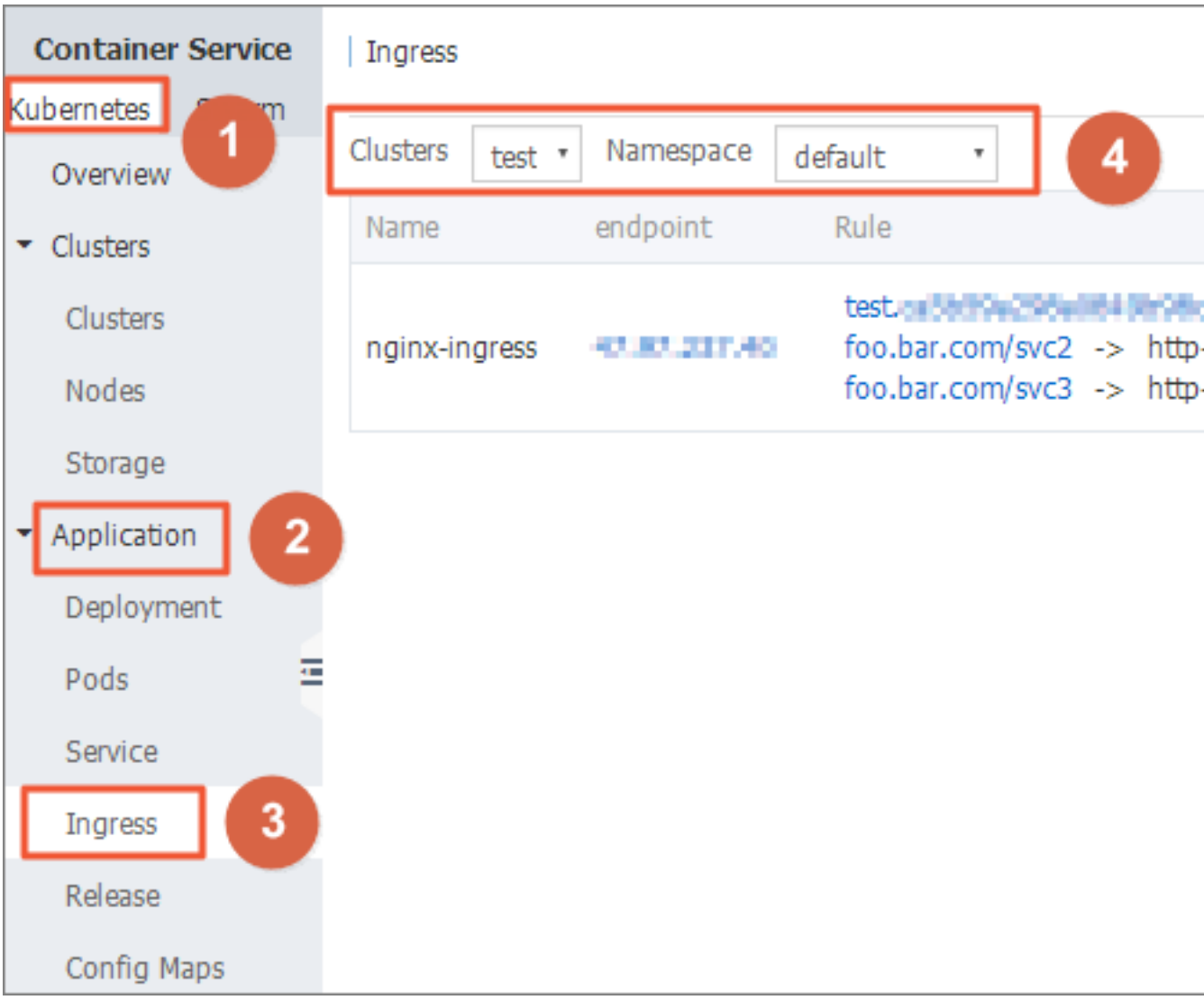
1.8.3 View Ingress details

Prerequisites

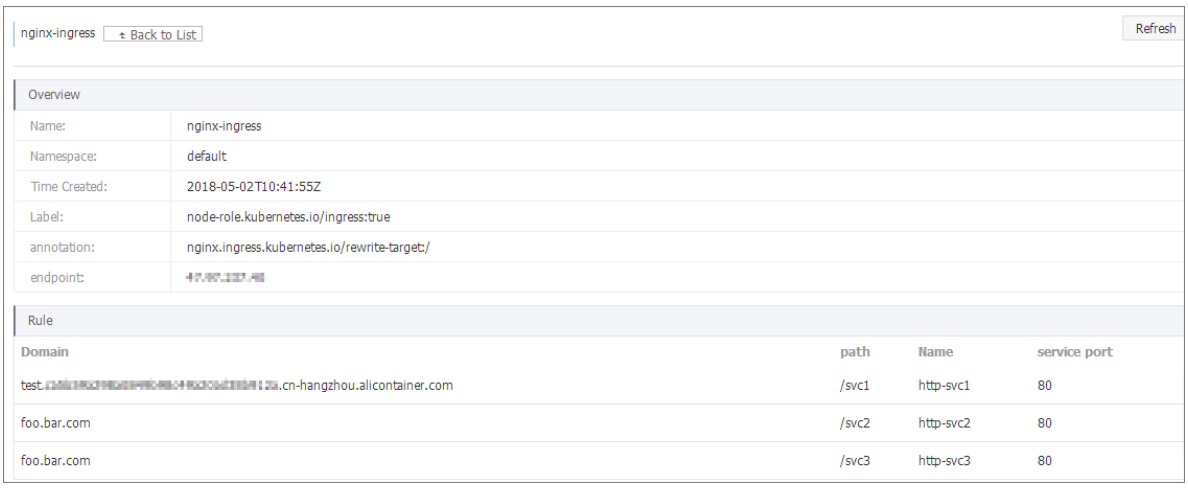
- You have successfully created a Kubernetes cluster and Ingress controller is running normally in the cluster. For how to create a Kubernetes cluster, see *Create a Kubernetes cluster*.
- You have successfully created an Ingress. For more information, see *Create an Ingress in Container Service console*.

Procedure

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



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



1.8.4 Update an Ingress

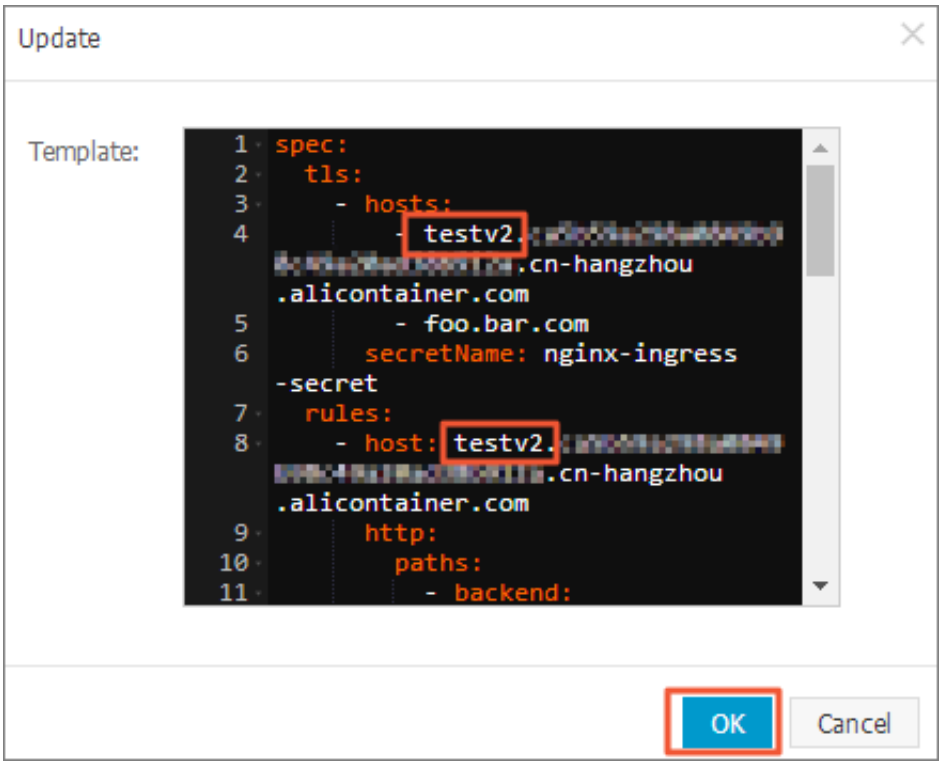
Prerequisites

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

Procedure

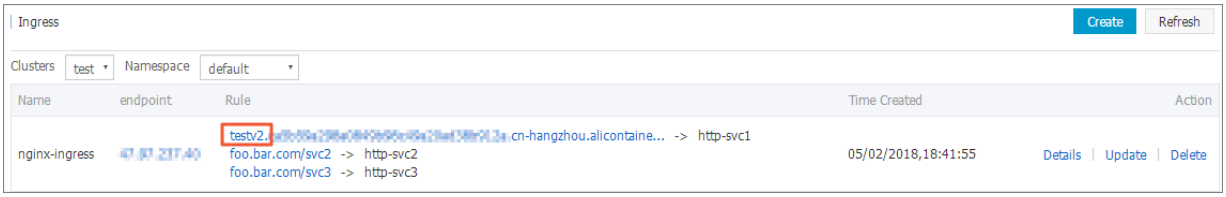
1. Log on to the *Container Service console*.
2. Click Kubernetes >**Application** > > **Ingress**in the left-side navigation pane.
3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click **Update** at the right of the Ingress.

4. Update the Ingress parameters in the displayed dialog box and then click **OK**. In this example, 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.



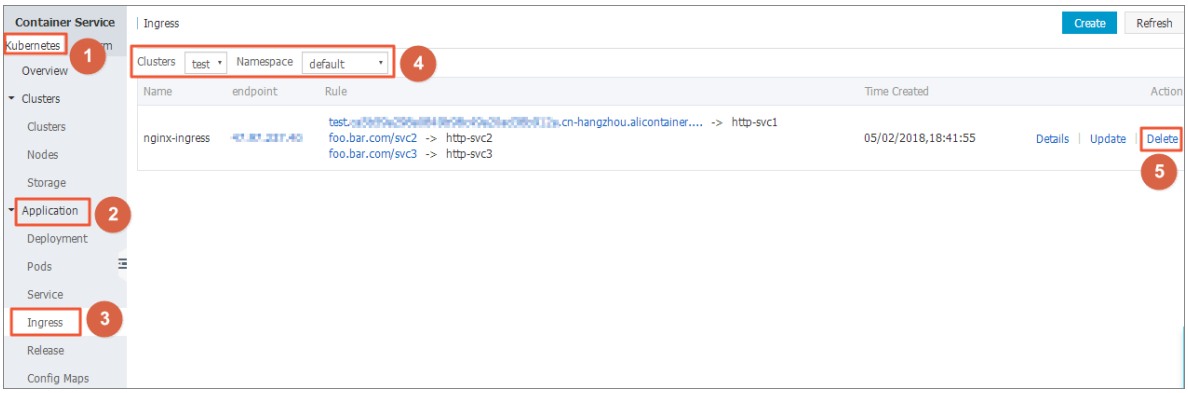
1.8.5 Delete an Ingress

Prerequisites

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

Procedure

1. Log on to the *Container Service console*.
2. 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 **Delete** at the right of the Ingress.



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

1.9 Manage a release

The Alibaba Cloud Kubernetes service enriches the cloud marketplace, in which the app catalog and service catalog functions integrate with the Helm package management tool and allow you to build the application in the cloud quickly. A chart can be released multiple times, which requires you to manage the release versions. Therefore, the Alibaba Cloud Kubernetes service provides the release function, which allows you to manage the applications released by using Helm in the Container Service console.

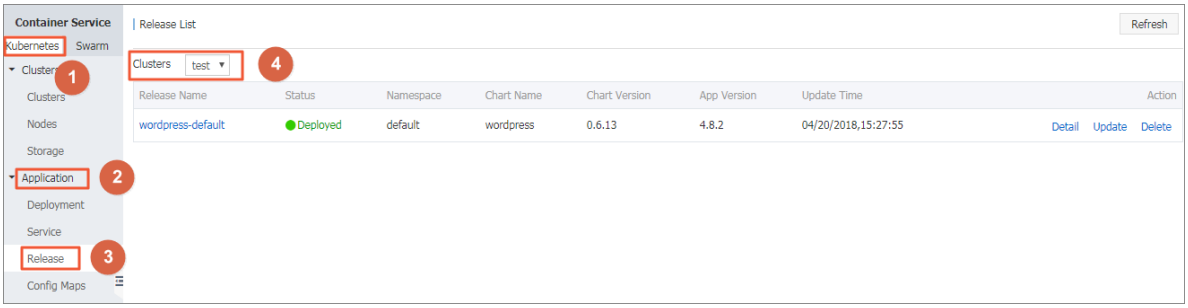
Prerequisites

- You have created a Kubernetes cluster successfully. For how to create a Kubernetes cluster, see *Create a Kubernetes cluster*.
- You have used the app catalog function or service catalog function to install a Helm application. For more information, see *Simplify Kubernetes application deployment by using Helm*. In this document, use the wordpress-default application as an example.

View release details

1. Log on to the *Container Service console*.
2. Click Kubernetes >**Application >** > **Release in the left-side navigation pane.**Select the cluster from the Clusters drop-down list.

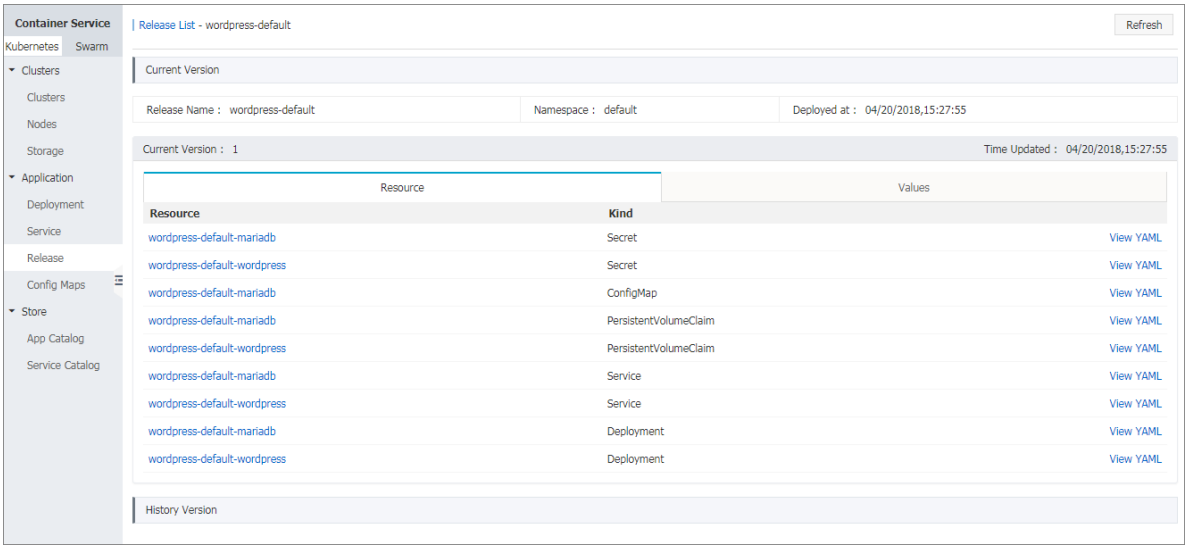
You can view the applications released by using the Helm package management tool and their services in the selected cluster.



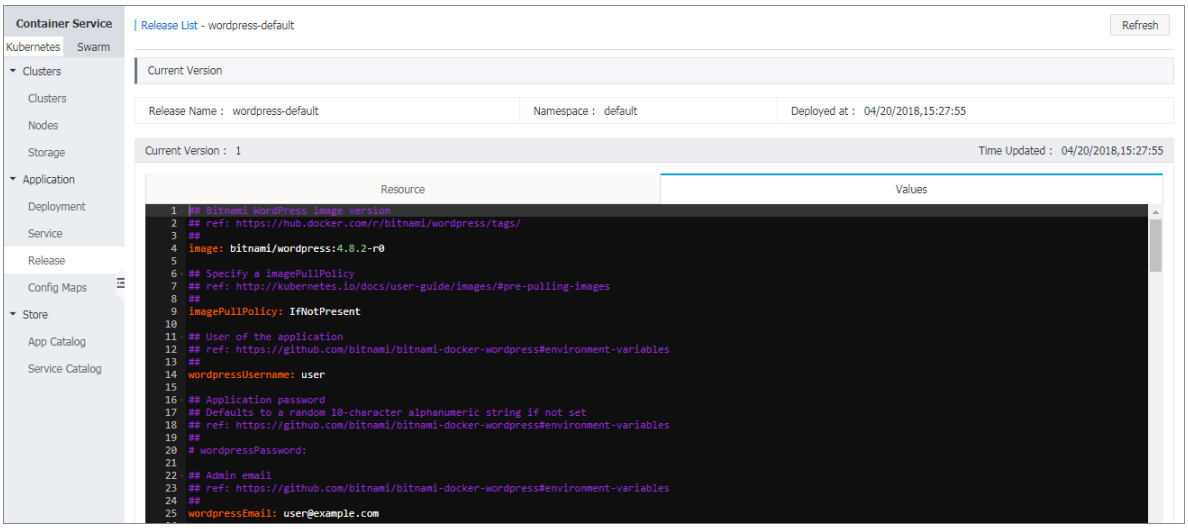
3. Take the wordpress-default as an example. Click **Detail** at the right of the release to view the release details,

   such as the current version and history version of this release (in this example, the current version is 1 and no history version exists). You can also view the resource information of wordpress-default, such as the resource name and resource type, and view the YAML information.

> **Note:**
> Click the resource name and you are redirected to the Kubernetes dashboard to view the detailed running status of this resource.



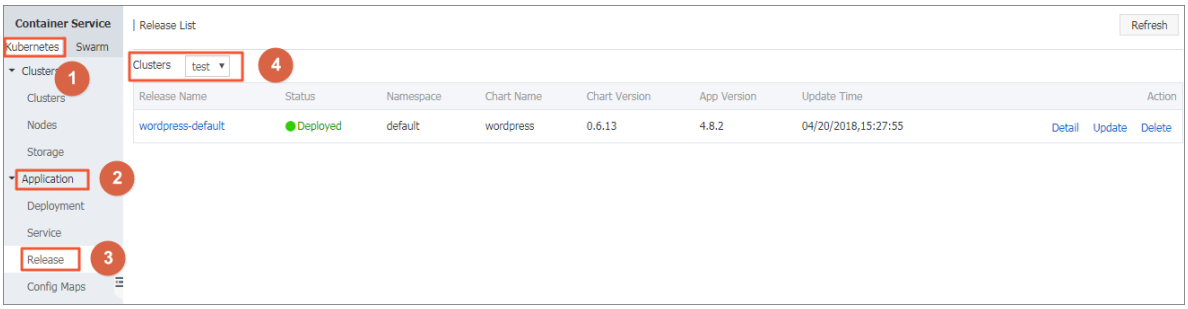
4. Click the **Values** tab to view the parameter configurations for installing the Helm package.



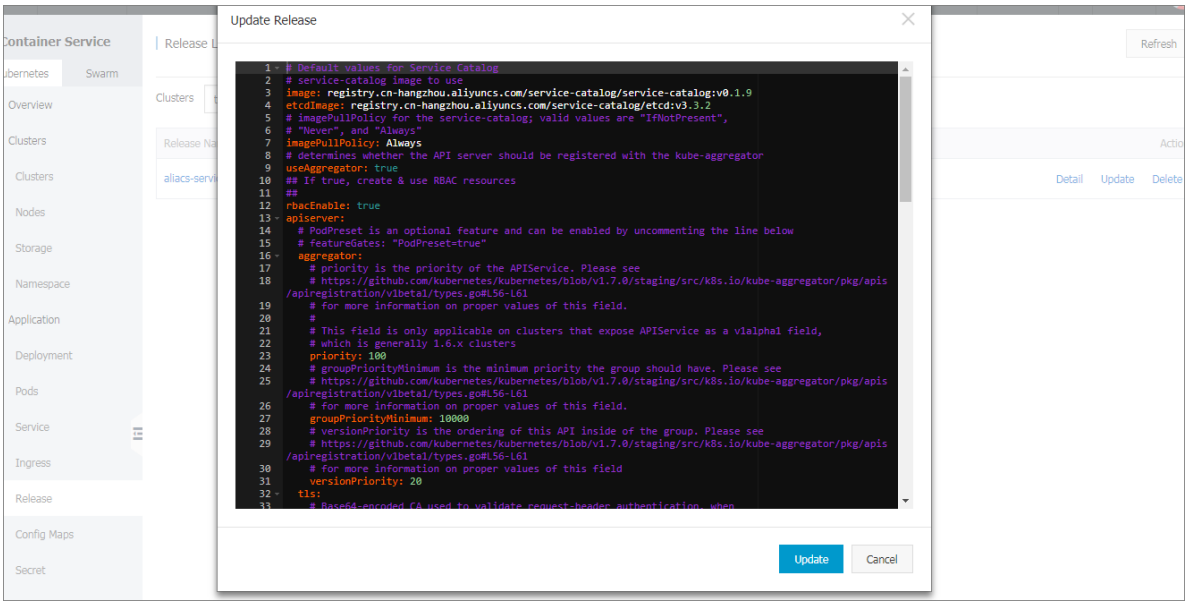
Update a release version

1. Log on to the *Container Service console*.
2. Click Kubernetes >**Application >** > **Release in the left-side navigation pane.**Select the cluster from the Clusters drop-down list.

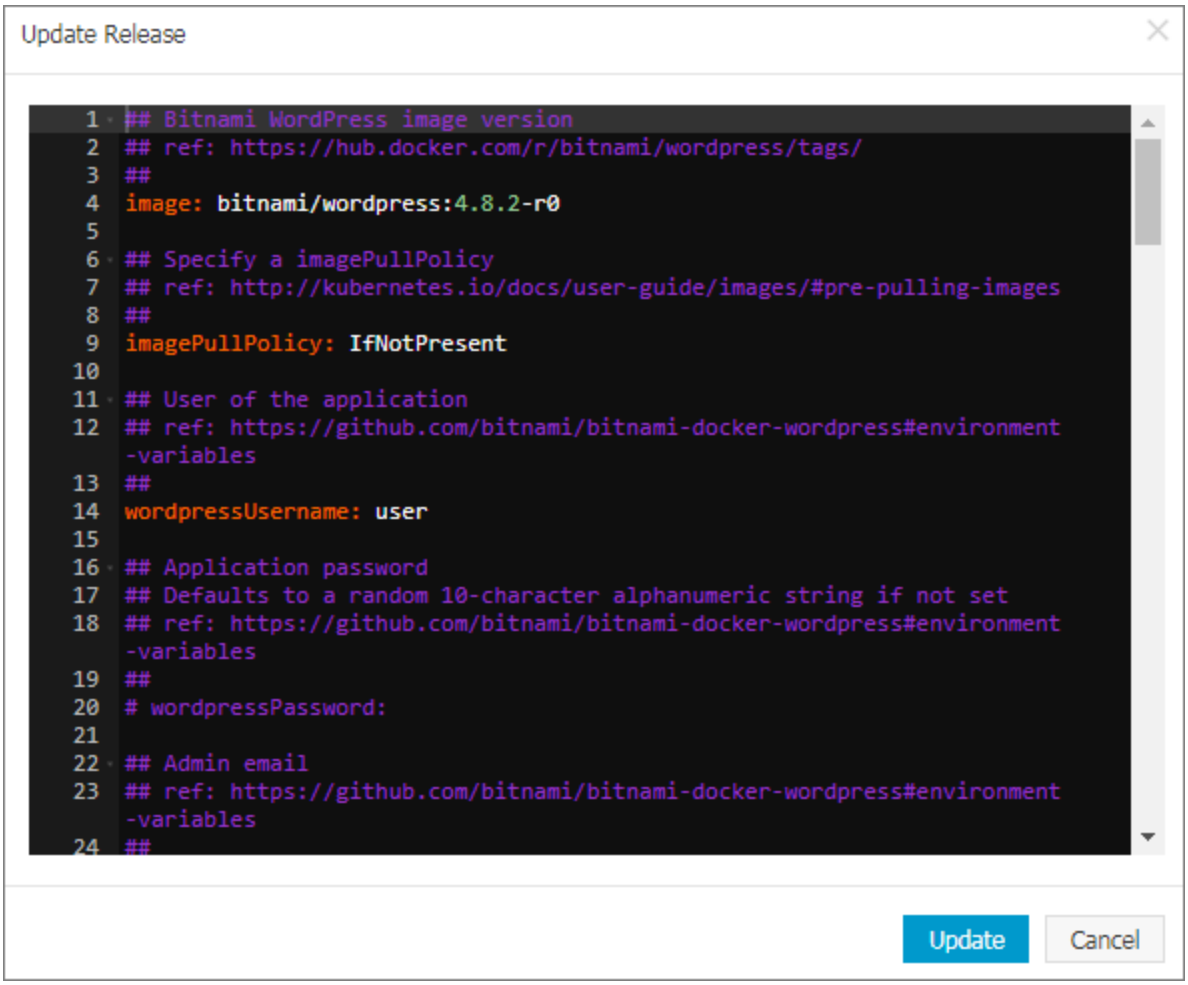
   You can view the applications released by using the Helm package management tool and their services in the selected cluster.



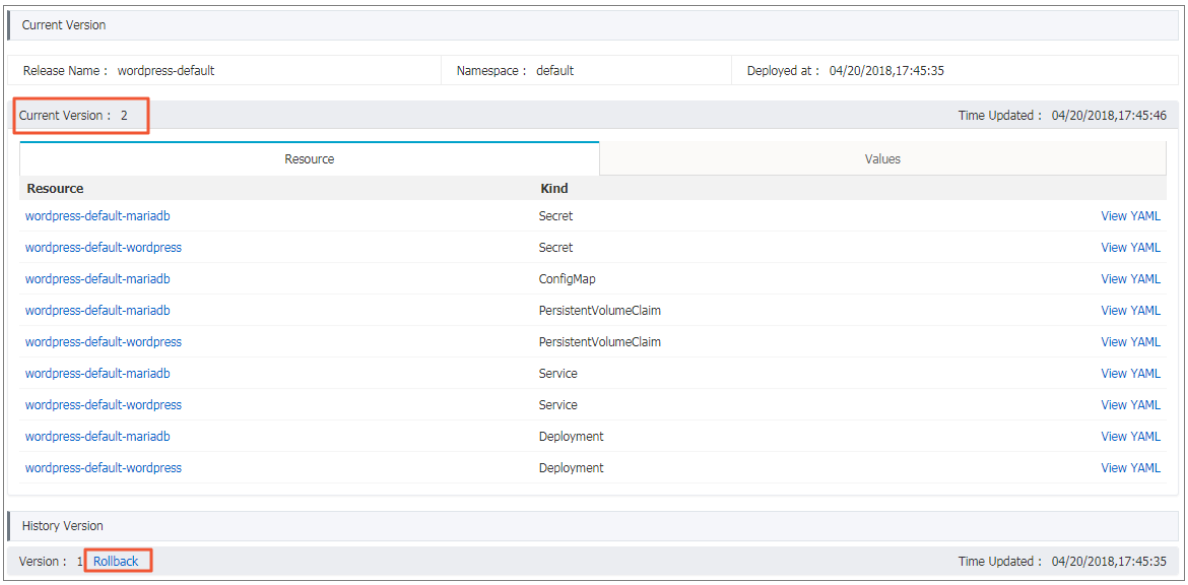
3. Take the wordpress-default as an example. Click **Update** at the right of the release to update the release. The Update Release dialog box appears.



4. Modify the parameters and then click **Update**.



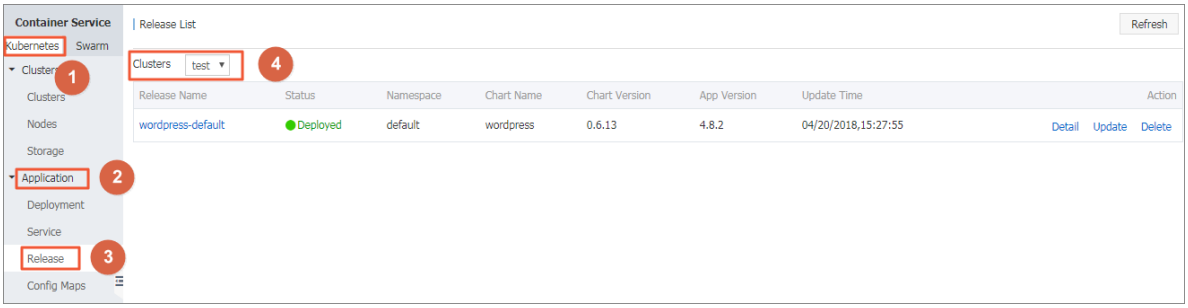
The current version is changed to 2 and you can find version 1 under History Version. To roll back the release, click **Rollback**.



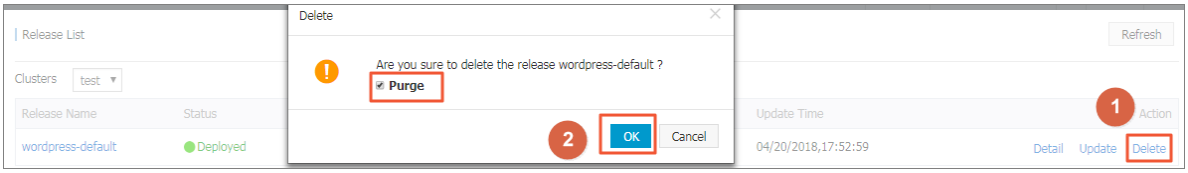
Delete a release

1. Log on to the *Container Service console*.
2. Click Kubernetes >**Application** > > **Release in the left-side navigation pane.**Select the cluster from the Clusters drop-down list.

   You can view the applications released by using the Helm package management tool and their services in the selected cluster.



3. Take the wordpress-default as an example. Click **Delete** at the right of the release to delete the release. The Delete dialog box appears.



4. Select the **Purge** check box to clear the release records if necessary. Click **OK** to delete the wordpress-default application and its resources such as the services and deployments.

1.10 App catalog

1.10.1 App catalog overview

Microservice is the theme of container era. The application microservice brings great challenge to the deployment and management. By dividing a large single application into several 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

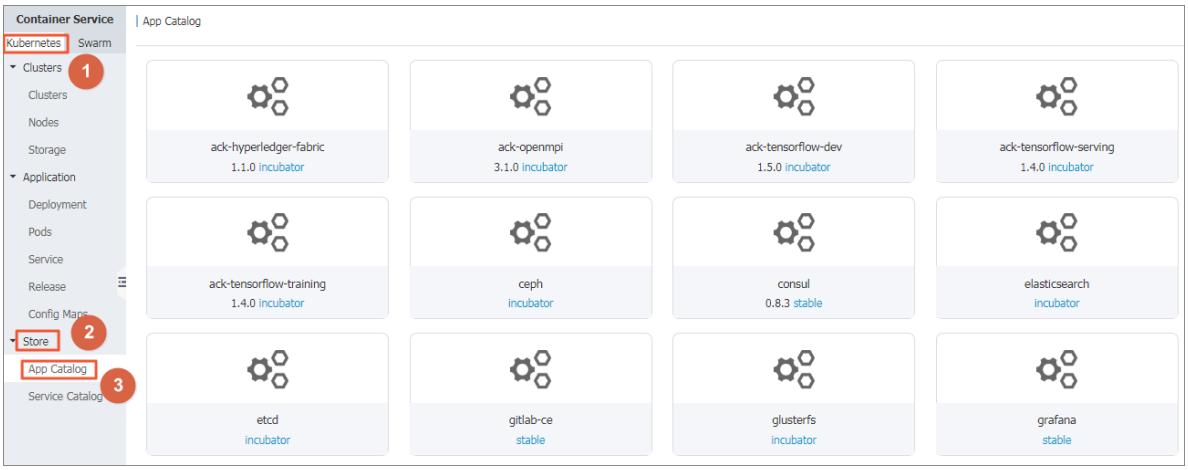
Currently, you can deploy and manage the charts in the app catalog by using the Helm tool. For more information, see *Simplify Kubernetes application deployment by using Helm*.

1.10.2 View app catalog list

Procedure

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

1.11 Plan Kubernetes CIDR blocks under VPC

Generally, you can select to create a Virtual Private Cloud (VPC) automatically and use the default network address when creating a Kubernetes cluster in Alibaba Cloud. In some complicated scenarios, plan the Elastic Compute Service (ECS) address, Kubernetes pod address, and Kubernetes service address on your own. This document introduces what the addresses in Kubernetes under Alibaba Cloud VPC environment are used for and how to plan the CIDR blocks.

Basic concepts of Kubernetes CIDR block

The concepts related to IP address are as follows:

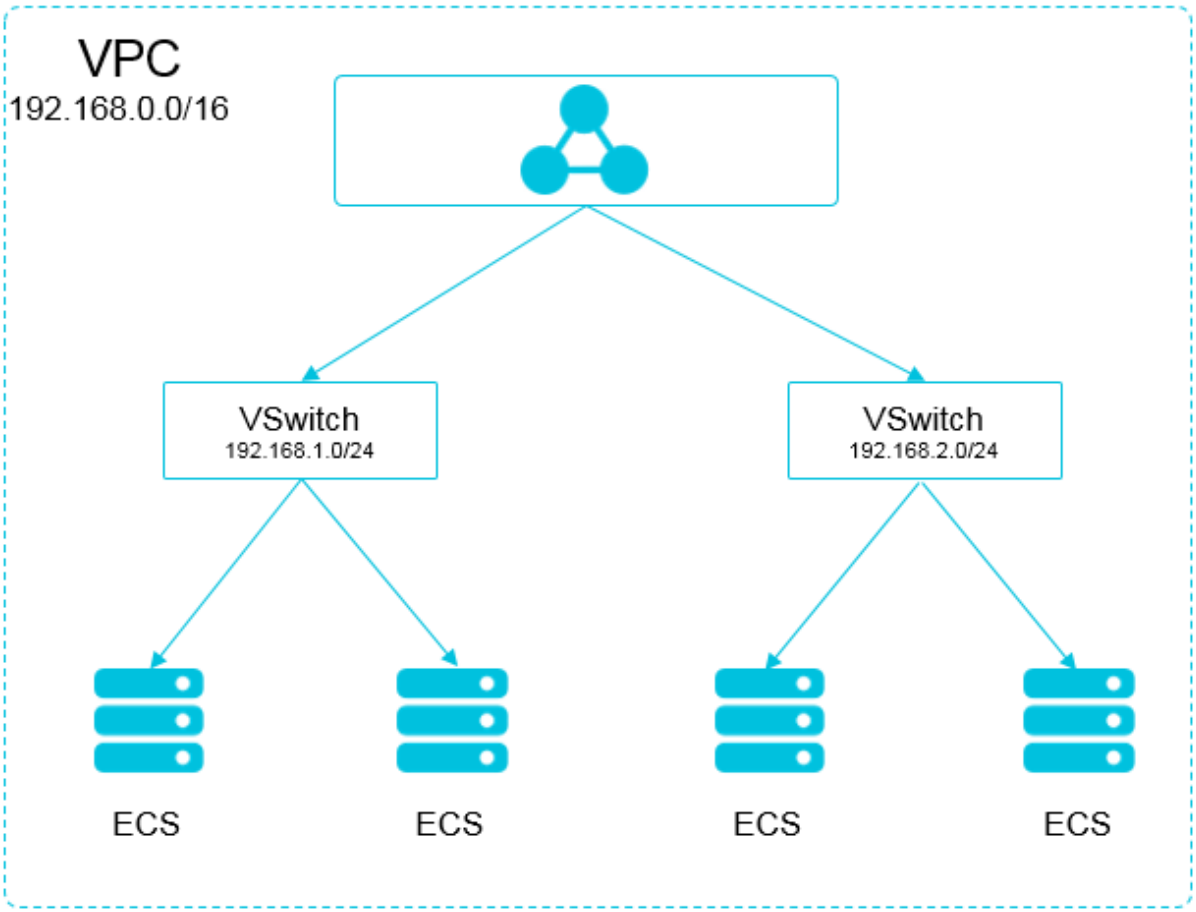
VPC CIDR block

The CIDR block selected when you create a VPC. Select the VPC CIDR block from 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.

VSwitch CIDR block

The CIDR block specified when you create a VSwitch in VPC. The VSwitch CIDR block must be the subset of the current VPC CIDR block, which can be the same as the VPC CIDR block but cannot go beyond that range. The address assigned to the ECS instance under the VSwitch is obtained from the VSwitch CIDR block. Multiple VSwitches can be created under one VPC, but the VSwitch CIDR blocks cannot overlap.

The VPC CIDR block structure is as follows.



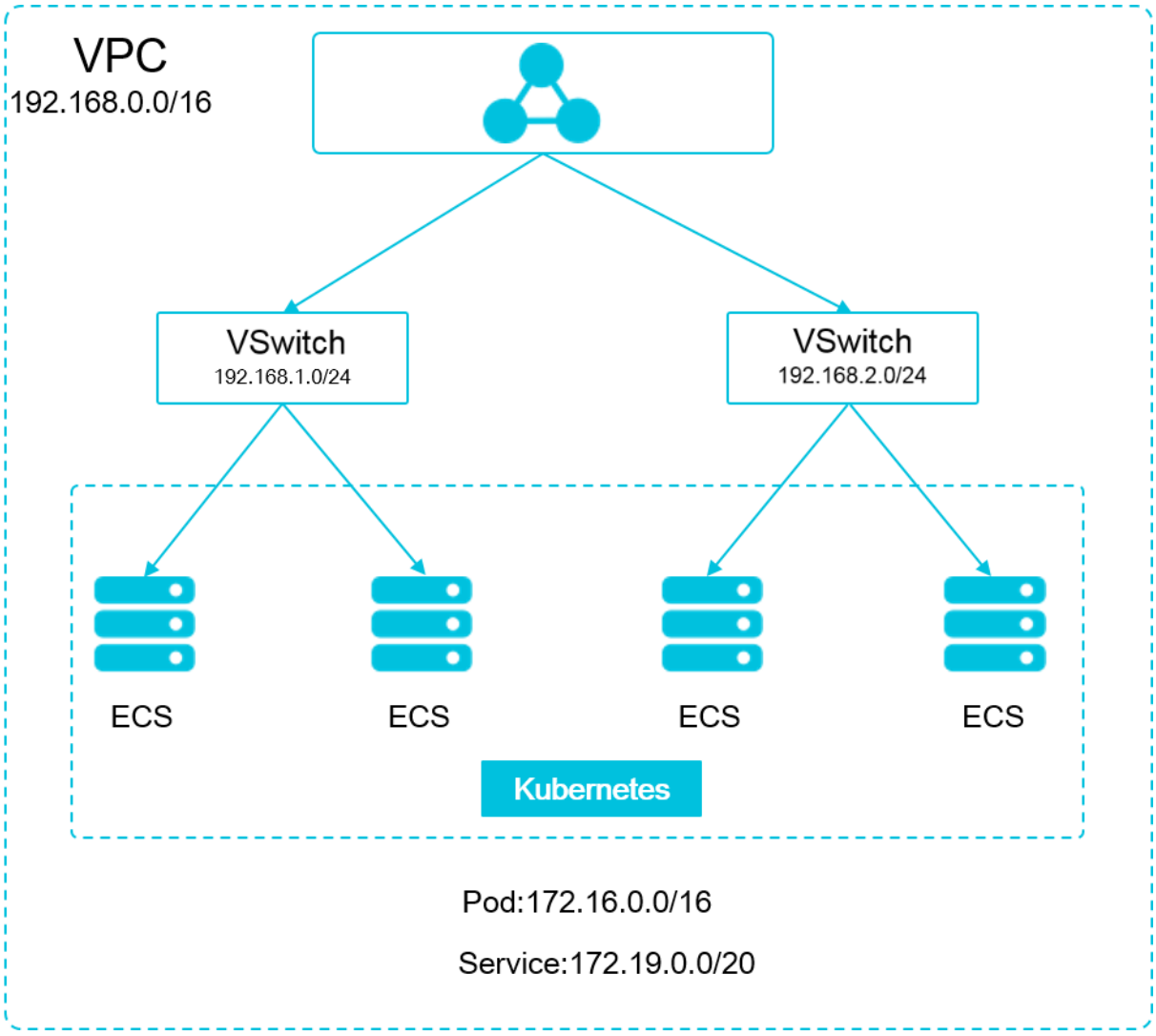
Pod CIDR block

Pod is a concept in Kubernetes. Each pod has one IP address. You can specify the pod CIDR block when creating a Kubernetes cluster in Alibaba Cloud Container Service and the pod CIDR block cannot overlap with the VPC CIDR block. For example, if the VPC CIDR block is 172.16.0.0/12, then the pod CIDR block of Kubernetes cannot use 172.16.0.0/16, 172.17.0.0/16, or any address that is included in 172.16.0.0/12.

Service CIDR block

Service is a concept in Kubernetes. Each service has its own address. The service CIDR block cannot overlap with the VPC CIDR block or pod CIDR block. The service address is only used in a Kubernetes cluster and cannot be used outside a Kubernetes cluster.

The relationship between Kubernetes CIDR block and VPC CIDR block is as follows.



How to select CIDR block

Scenario of one VPC and one Kubernetes cluster

This is the simplest scenario. The VPC address is determined when the VPC is created. Select a CIDR block different from that of the current VPC when creating a Kubernetes cluster.

Scenario of one VPC and multiple Kubernetes clusters

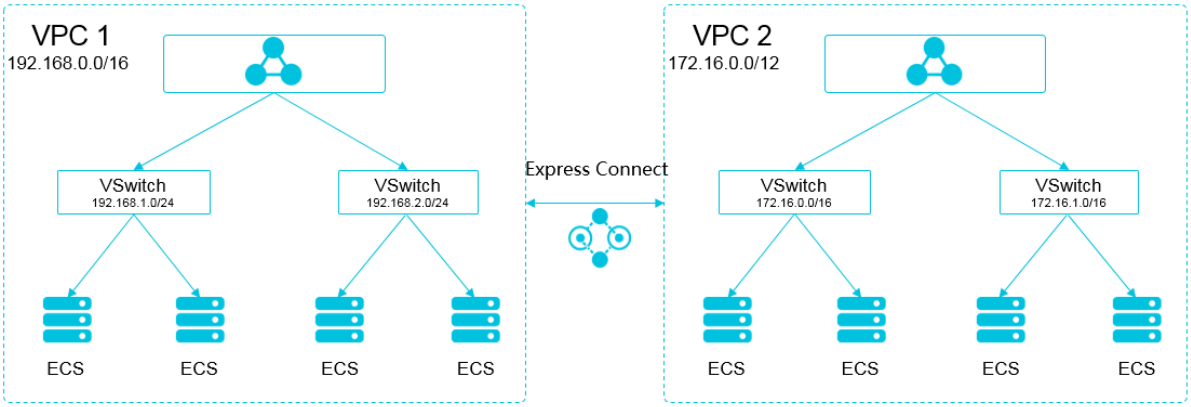
Create multiple Kubernetes clusters under one VPC. In the default network mode (Flannel), the pod message needs to be routed by using VPC, and Container Service automatically configures the route table to each pod CIDR block on the VPC route. The pod CIDR blocks of all the Kubernetes clusters cannot overlap, but the service CIDR blocks can overlap.

The VPC address is determined when the VPC is created. Select a CIDR block that does not overlap with the VPC address or other pod CIDR blocks for each Kubernetes cluster when creating a Kubernetes cluster.

In such a situation, parts of the Kubernetes clusters are interconnected. The pod of one Kubernetes cluster can directly access the pod and ECS instance of another Kubernetes cluster, but cannot access the service of another Kubernetes cluster.

Scenario of VPC interconnection

You can configure what messages are to be sent to the opposite VPC by using route tables when two VPCs are interconnected. Take the following scenario as an example: VPC 1 uses the CIDR block 192.168.0.0/16 and VPC 2 uses the CIDR block 172.16.0.0/12. By using route tables, specify to send the messages of 172.16.0.0/12 in VPC 1 to VPC 2.



In such a situation, the CIDR block of the Kubernetes cluster created in VPC 1 cannot overlap with VPC 1 CIDR block or the CIDR block to be routed to VPC 2. The same applies to the scenario when you create a Kubernetes cluster in VPC 2. In this example, the pod CIDR block of the Kubernetes cluster can select a sub-segment under 10.0.0.0/8.

Note:

The CIDR block routing to VPC 2 can be considered as an occupied address. Kubernetes clusters cannot overlap with an occupied address.

To access the Kubernetes pod of VPC 1 in VPC 2, configure the route to the Kubernetes cluster in VPC 2.

Scenario of VPC to IDC

Similar to the scenario of VPC interconnection, if parts of the CIDR blocks in VPC route to IDC, the pod address of Kubernetes clusters cannot overlap with those addresses. To access the pod address of Kubernetes clusters in IDC, configure the route table to leased line virtual border router (VBR) in IDC.

1.12 Server Load Balancer

1.12.1 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.

1.12.2 Access services by using Server Load Balancer

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

Operate by using command lines

1. Create an Nginx application by using command lines.

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

2. Create Alibaba Cloud Server Load Balancer service for the Nginx application and specify `type=LoadBalancer` to expose the Nginx service to the Internet.

```
root@master # kubectl expose deployment nginx --port=80 --target-
port=80 --type=LoadBalancer
root@master # kubectl get svc
NAME CLUSTER-IP EXTERNAL-IP PORT(S) AGE
nginx 172.19.10.209 101.37.192.20 80:31891/TCP 4s
```

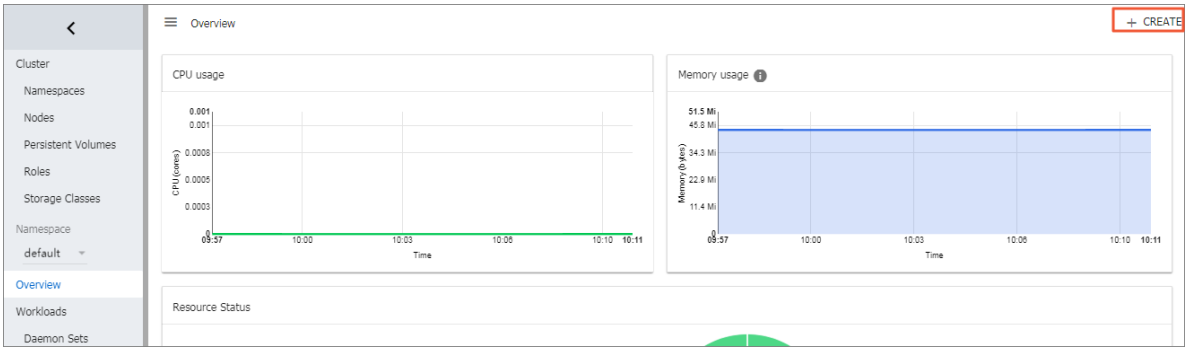
3. Visit `http://101.37.192.20` in the browser to access your Nginx service.

Operate by using Kubernetes dashboard

1. Save the following yml codes 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
```

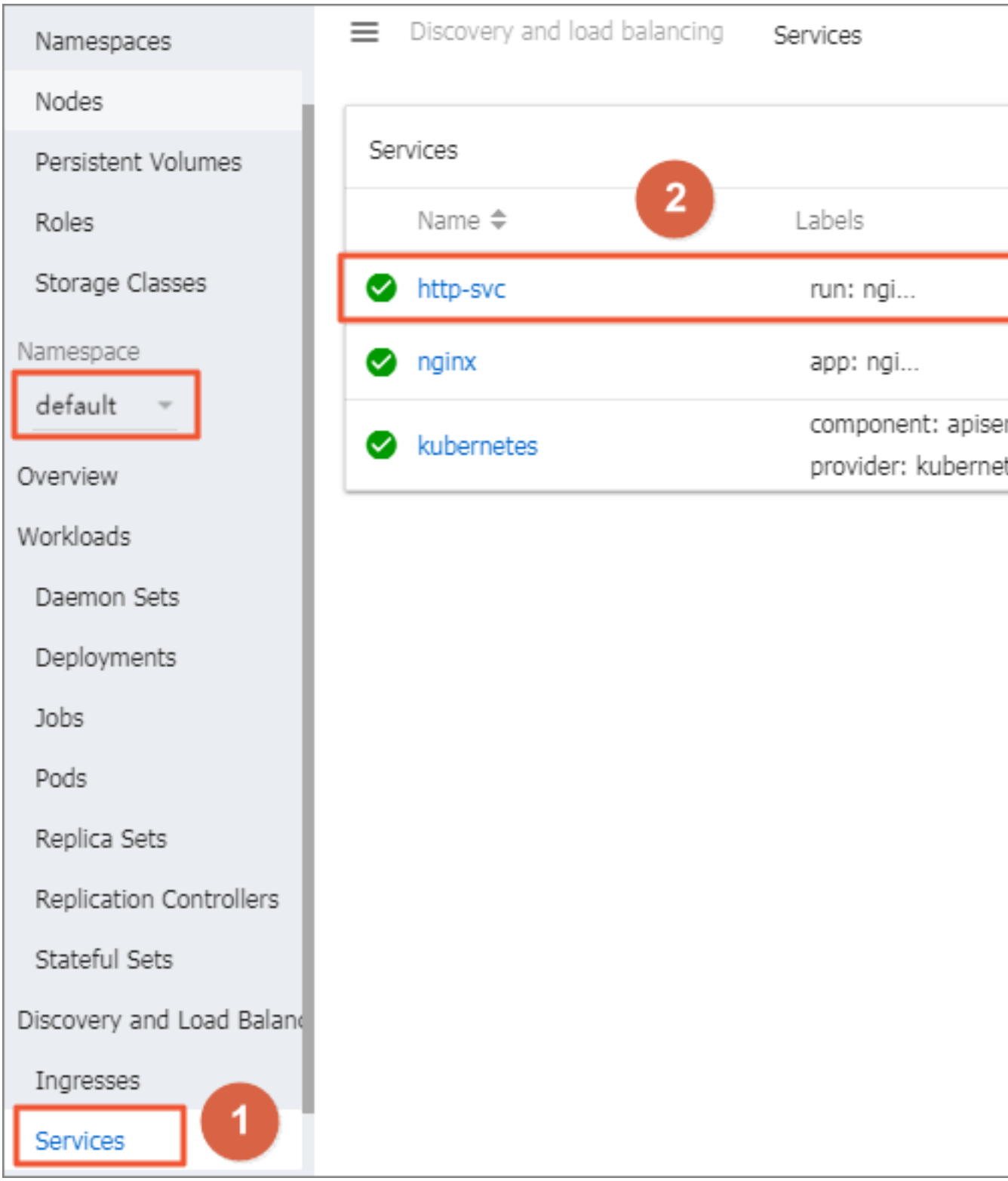
2. Log on to the *Container Service console*. Under Kubernetes, click Clusters in the left-side navigation pane. Click **Dashboard** at the right of the cluster to enter the Kubernetes dashboard.
3. Click **CREATE** in the upper-right corner to create an application.



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

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

   You can view the created Nginx service `http-svc` and the Server Load Balancer address `http://120.55.179.145:80`.



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

More information

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

Annotations

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

Use existing intranet Server Load Balancer instance

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

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

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

Create an HTTPS type Server Load Balancer instance

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

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/alicloud-loadbalancer-cert-id: your-
cert-id
    service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port: "
https:443"
  labels:
    run: nginx
```

```
  name: nginx
  namespace: default
spec:
  ports:
  - name: web
    port: 443
    protocol: TCP
    targetPort: 443
  selector:
    run: nginx
  sessionAffinity: None
  type: LoadBalancer
```

Note:

Annotations are case sensitive.

Annotation	Description	Default value
service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port	Use commas (,) to separate multiple values. For example, https:443,http:80.	None
service.beta.kubernetes.io/alicloud-loadbalancer-address-type	The value is Internet or intranet.	Internet
service.beta.kubernetes.io/alicloud-loadbalancer-slb-network-type	Server Load Balancer network type. The value is classic or VPC.	classic
service.beta.kubernetes.io/alicloud-loadbalancer-charge-type	The value is paybytraffic or paybybandwidth.	paybybandwidth
service.beta.kubernetes.io/alicloud-loadbalancer-id	The Server Load Balancer instance ID. Use loadbalancer-id to specify an existing Server Load Balancer instance. The existing listener will be overwritten. The Server Load Balancer instance will not be deleted when the service is deleted.	None
service.beta.kubernetes.io/alicloud-loadbalancer-backend-label	Use label to specify what nodes are mounted to the Server Load Balancer backend.	None

Annotation	Description	Default value
service.beta.kubernetes.io/alicloud-loadbalancer-region	The region in which Server Load Balancer resides.	None
service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth	Server Load Balancer bandwidth.	50
service.beta.kubernetes.io/alicloud-loadbalancer-cert-id	Authentication ID on Alibaba Cloud. Upload the certificate first.	“”
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-flag	The value is on or off.	The default value is off. No need to modify the TCP parameters because TCP enables health check by default and you cannot configure it.
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-type	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-uri	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-port	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-healthy-threshold	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-unhealthy-threshold	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval	See *HealthCheck*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout	See *HealthCheck*.	

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

1.12.3 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: |

  creationTimestamp: 2018-03-20T07:10:18Z
  labels:
    app: ingress-nginx
  name: nginx-configuration
  namespace: kube-system
  selfLink: /api/v1/namespaces/kube-system/configmaps/nginx-
configuration
```

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

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

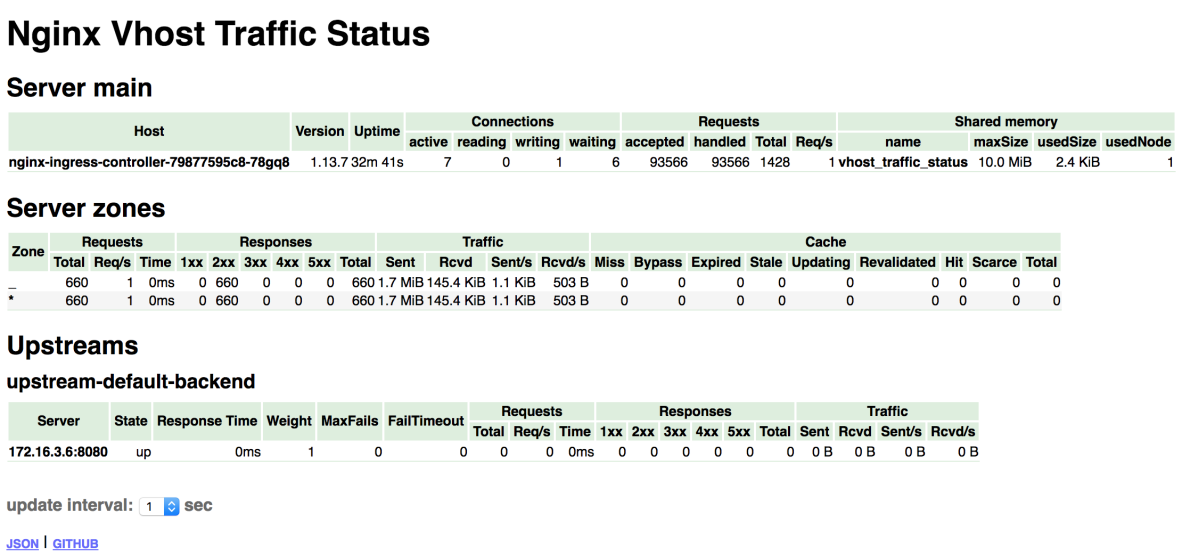
3. Locally access the Ingress Nginx monitoring console.

Note:

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

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

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



Nginx Vhost Traffic Status

Server main

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

Server zones

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

Upstreams

upstream-default-backend

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

update interval: 1 sec

JSON | GITHUB

Enable VTS module by using the Kubernetes dashboard

1. Log on to the *Container Service console*.
2. On the Cluster List page of Kubernetes clusters, click **Dashboard** at the right of a cluster to enter the Kubernetes dashboard page.
3. Select kube-system under Namespace in the left-side navigation pane. Click Config Maps in the left-side navigation pane. Click the icon at the right of nginx-configuration and then select View/edit YAML. Edit the config map to add the configuration item `enable-vts-status: "true"`.

   The contents of the saved Ingress ConfigMap are as follows:

```
  "kind": "ConfigMap",
  "apiVersion": "v1",
  "metadata": {
    "name": "nginx-configuration",
    "namespace": "kube-system",
    "selfLink": "/api/v1/namespaces/kube-system/configmaps/nginx-
configuration",
    "creationTimestamp": "2018-03-20T07:10:18Z",
    "labels": {
      "app": "ingress-nginx"
```

```
    "annotations": {
      "kubectl.kubernetes.io/last-applied-configuration": "{\"
apiVersion\":\"v1\",\"data\":{\"proxy-body-size\":\"20m\"},\"kind\":
\"ConfigMap\",\"metadata\":{\"annotations\":{},\"labels\":{\"app\":
\"ingress-nginx\"},\"name\":\"nginx-configuration\",\"namespace\":\"
kube-system\"}}\n"


 "data": {
   "proxy-body-size": "20m",
   "enable-vts-status": "true"
```

4. Locally access the Ingress Nginx monitoring console.

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

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

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

Nginx Vhost Traffic Status

Server main

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

Server zones

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

Upstreams

upstream-default-backend

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

update interval: 1 sec

JSON | GITHUB

1.12.4 Support for Ingress

In Kubernetes clusters, Ingress is a collection of rules that authorize inbound connection to the cluster services and provides you with Layer-7 Server Load Balancer capabilities. You can provide the Ingress configuration with externally accessible URL, Server Load Balancer, SSL, and name-based virtual host.

Prerequisites

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

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

root@master # kubectl expose deploy nginx --name=http-svc --port=80 --
target-port=80
root@master # kubectl expose deploy nginx --name=http-svc1 --port=80
 --target-port=80
root@master # kubectl expose deploy nginx --name=http-svc2 --port=80
 --target-port80
root@master # kubectl expose deploy nginx --name=http-svc3 --port=80
 --target-port=80
```

Simple routing service

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

```
root@master # cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /svc
        backend:
          serviceName: http-svc
          servicePort: 80
EOF
root@master # kubectl get ing
NAME HOSTS ADDRESS PORTS AGE
simple * 101.37.192.211 80 11s
```

Now visit `http://101.37.192.211/svc` to access the Nginx service.

Simple fanout routing based on domain names

If you have multiple domain names providing different external services, you can generate the following configuration to implement a simple fanout effect based on domain names:

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

```
metadata:
  name: simple-fanout
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: http-svc1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: http-svc2
          servicePort: 80
  - host: foo.example.com
    http:
      paths:
      - path: /film
        backend:
          serviceName: http-svc3
          servicePort: 80
EOF
root@master # kubectl get ing
NAME HOSTS ADDRESS PORTS AGE
simple-fanout * 101.37.192.211 80 11s
```

Then, you can access the `http-svc1` service by using `http://foo.bar.com/foo`, access the `http-svc2` service by using `http://foo.bar.com/bar`, and access the `http-svc3` service by using `http://foo.example.com/film`.

Note:

- In a production environment, point the domain name to the preceding returned address `101.37.192.211`.
- In a test environment, 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"
   ```

   This command generates a certificate file *tls.crt* and a private key file *tls.key*.

Create a Kubernetes secret named *foo.bar* by using the certificate and private key.

Reference this secret when you create the Ingress service.

```
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, you are automatically redirected to `https://foo.bar.com/foo` after accessing `http://foo.bar.com/foo`.

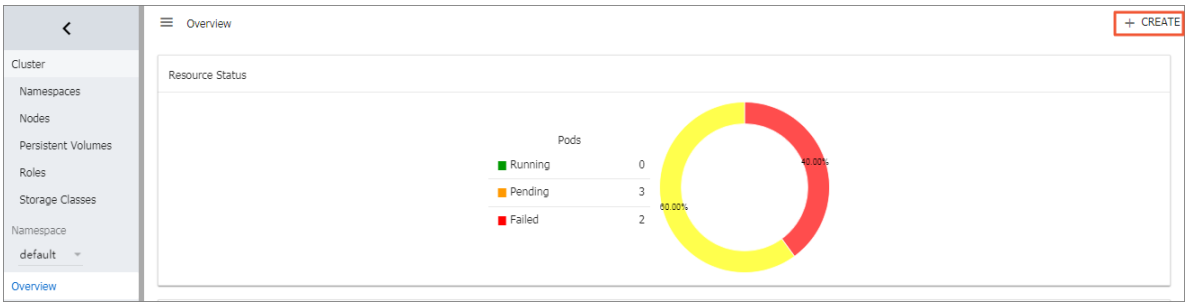
Deploy Ingress in Kubernetes dashboard

1. Save the following yml codes 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
```

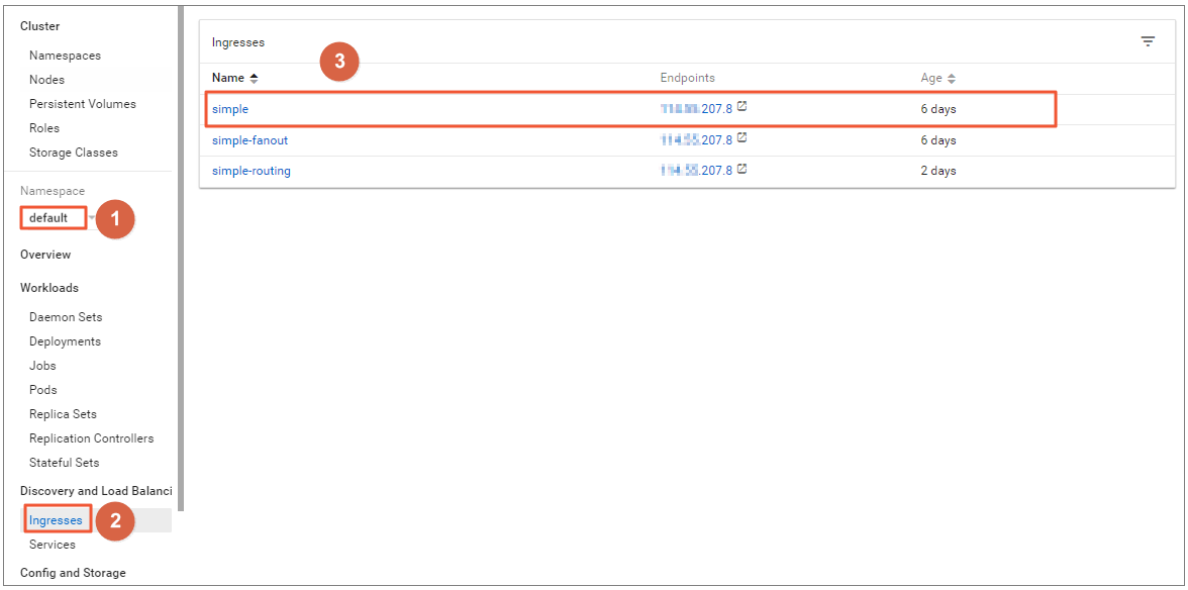
2. Log on to the *Container Service console*. Under Kubernetes, click Clusters in the left-side navigation pane. Click **Dashboard** at the right of the cluster to enter the Kubernetes dashboard.
3. Click **CREATE** in the upper-right corner to create an application.



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

   Then, an Ingress Layer-7 proxy route is created on 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://114.55.207.8/svc`.



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

1.13 Storage

1.13.1 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.

1.13.2 Use Alibaba Cloud NAS

You can use the Alibaba Cloud NAS data volumes in Container Service Kubernetes clusters.

Currently, Alibaba Cloud NAS provides the following two Kubernetes mount methods:

- *Static storage volumes*

  You can use the static storage volumes by:

  - Using the flexvolume plug-in.

- Using the volume method.
- Using PV/PVC.
- Using NFS drive of Kubernetes.

- *Dynamic storage volumes*

Prerequisite

Before using NAS data volumes, you must create a file system in the NAS console and add the mount point of a Kubernetes cluster in the file system. The created NAS file system and your cluster must be in the same Virtual Private Cloud (VPC).

Static storage volumes

You can use Alibaba Cloud NAS file storage service by using the flexvolume plug-in provided by Alibaba Cloud or the NFS drive of Kubernetes.

Use flexvolume plug-in

Use the flexvolume plug-in and then you can use the Alibaba Cloud NAS data volumes by using the volume method or using PV/PVC.

Note:

- NAS is a shared storage and can provide shared storage service for multiple pods at the same time.
- server: The mount point of the NAS data disk.
- path: The mount directory for connecting to the NAS data volumes. You can mount NAS data volumes to a NAS sub-directory. The system automatically creates the sub-directory if the sub-directory does not exist and mounts the NAS data volumes to the created sub-directory.
- vers: Defines the version number of NFS mount protocol. 4.0 is supported.
- mode: Defines the access permission of the mount directory. The mount permission cannot be configured if you mount the NAS data volumes to the NAS root directory. If the NAS disk contains a huge amount of data, configuring the mode leads to the slow mounting or even the mounting failure.

Using the volume method.

Use the `nas-deploy.yaml` file to create the pod.

```
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 PV/PVC

Step 1 Create PV

You can create NAS data volumes in the Container Service console or by using the YAML file.

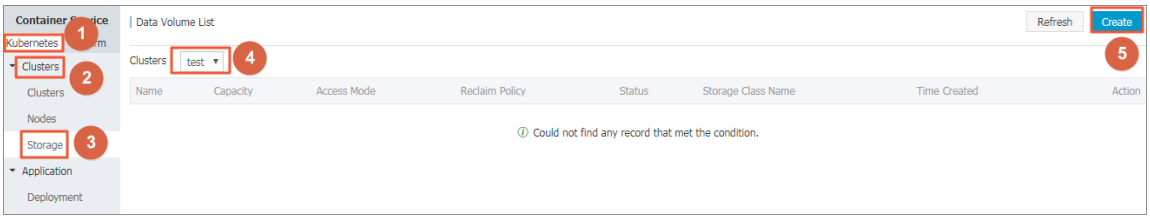
- Create PV by using YAML file

  Use the `nas-pv.yaml` file to create the PV.

  ```
  apiVersion: v1
  kind: PersistentVolume
  metadata:
    name: pv-nas
  spec:
    capacity:
      storage: 5Gi
    storageClassName: nas
    accessModes:
      - ReadWriteMany
    flexVolume:
      driver: "alicloud/nas"
      options:
        server: "0cd8b4a576-uih75.cn-hangzhou.nas.aliyuncs.com"
        path: "/k8s"
        vers: "4.0"
  ```

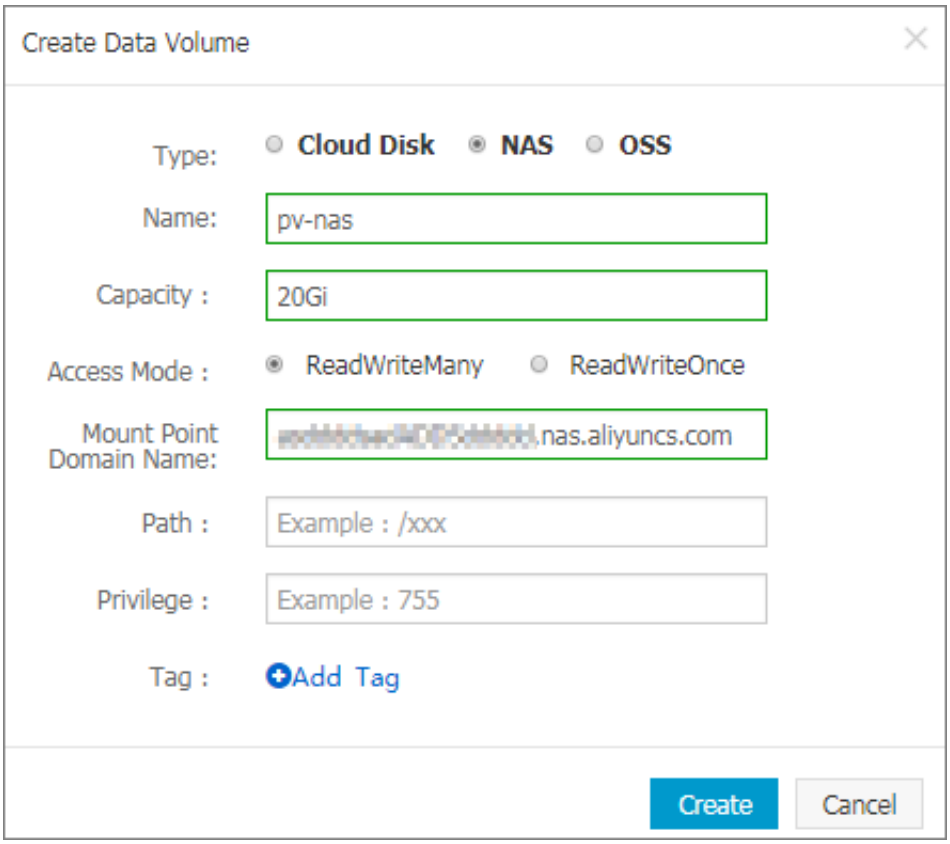
- Create NAS data volumes in Container Service console

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



4. The Create Data Volume dialog box appears. Configure the data volume parameters.

   - **Type:** Select NAS in this example.
   - **Name**: Enter the name of the data volume you are about to create. The data volume name must be unique in the cluster. In this example, enter pv-nas.
   - **Capacity**: Enter the capacity of the data volume to be created. Make sure the capacity cannot exceed the disk capacity.
   - **Access Mode**: ReadWriteMany is selected by default.
   - **Mount Point Domain Name**: Enter the mount address of the mount point in the NAS file system for the cluster.
   - **Path**: The sub-directory under the NAS path, which starts with a forward slash (`/`). The data volume is mounted to the specified sub-directory after being created.
     - If this sub-directory does not exist in the NAS root directory, the data volume is mounted after the sub-directory is created by default.
     - If this field is left empty, the data volume is mounted to the NAS root directory by default.
   - **Privilege**: Configure the access permission of the mount directory, such as 755, 644, and 777.
     - You can only configure the privilege when the data volume is mounted to the NAS sub -directory, that is, you cannot configure the privilege if the data volume is mounted to the NAS root directory.
     - If this field is left empty, use the permissions of the NAS files by default.
   - **Tag**: Click Add Tag to add tags for this data volume.



5. Click **Create** after the configurations.

Step 2 Create PVC

Use the *nas-pvc.yaml* file to create the PVC.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-nas
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: nas
  resources:
    requests:
      storage: 5Gi
```

Step 3 Create pod

Use the *nas-pod.yaml* file to create the pod.

```
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
```

Using NFS drive of Kubernetes.

Step 1 Create a NAS file system

Log on to the *NAS console* to create a NAS file system.

Note:
The created NAS file system and your cluster must be in the same region.

Assume that your mount point is `055f84ad83-ixxxx.cn-hangzhou.nas.aliyuncs.com`.

Step 2 Create PV

You can create NAS data volumes in the Container Service console or by using an orchestration template.

- **Use an orchestration template**

  Use the *nas-pv.yaml* file to create the PV.

  Run the following commands to create a NAS type PersistentVolume.

```
root@master # cat << EOF |kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata:
   name: nas
spec:
   capacity:
     storage: 8Gi
   accessModes:
     - ReadWriteMany
   persistentVolumeReclaimPolicy: Retain
   nfs:
     path: /
     server: 055f84ad83-ixxxx.cn-hangzhou.nas.aliyuncs.com
EOF
```

- **Create NAS data volumes in Container Service console**

  For more information, see Create NAS data volumes in Container Service console in *Use PV/PVC*.

Step 2 Create PVC

Create a PersistentVolumeClaim to request to bind this PersistentVolume.

```
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 pod

Create an application to declare to mount and use this data 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
```

Then, the NAS remote file system is mounted to your pod application.

Dynamic storage volumes

To use dynamic NAS storage volumes, you must manually install the drive plug-in and configure the NAS mount point.

Install the plug-in

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: alicloud-nas
provisioner: alicloud/nas
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: alicloud-nas-controller
  namespace: kube-system
```

```
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: run-alicloud-nas-controller
subjects:
  - kind: ServiceAccount
    name: alicloud-nas-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-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: alicloud-nas-controller
      containers:
        - name: alicloud-nas-controller
          image: registry.cn-hangzhou.aliyuncs.com/acs/alicloud-nas-
controller:v1.8.4
          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
          nfs:
            server: 0cd8b4a576-mmi32.cn-hangzhou.nas.aliyuncs.com
            path: /
```

Use dynamic storage volumes

```
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
```

1.13.3 Use Alibaba Cloud OSS

You can use the Alibaba Cloud Object Storage Service (OSS) data volumes in Alibaba Cloud Container Service Kubernetes clusters.

Currently, OSS static storage volumes are supported, while OSS dynamic storage volumes are not supported. You can use the OSS static storage volumes by:

- Using the volume method.
- Using PV/PVC.

Prerequisites

You must create a bucket in the OSS console before using the OSS static storage volumes.

Instructions

- OSS is a shared storage and can provide shared storage service for multiple pods at the same time.
- bucket: Currently, Container Service only supports mounting buckets and cannot mount the sub-directories or files under the bucket.
- url: The OSS endpoint, which is the access domain name for mounting OSS.
- akId: Your AccessKey ID.

- akSecret: Your AccessKey Secret.
- otherOpts: Customized parameter input in the format of `-o *** -o ***` is supported when mounting OSS.

Note

If your Kubernetes cluster is created before Feb 6th, 2018, *Install the plug-in* before using the data volumes. To use OSS data volumes, you must create the secret and enter the AccessKey information when deploying the flexvolume service.

Use OSS static storage volumes

Use volume method

Use the *oss-deploy.yaml* file to create the pod.

```
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"
      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 PV/PVC (currently, dynamic pv is not supported)

Step 1 Create PV

You can create the PV in the Container Service console or by using the YAML file.

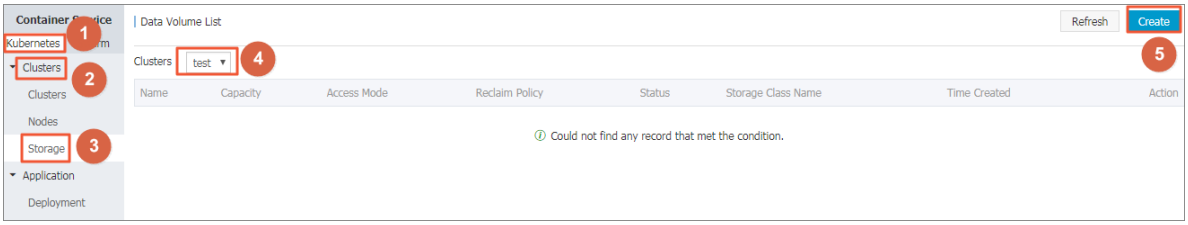
Create PV by using YAML file

Use the *oss-pv.yaml* file to create the PV.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-oss
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  storageClassName: oss
  flexVolume:
    driver: "alicloud/oss"
    options:
      bucket: "docker"
      url: "oss-cn-hangzhou.aliyuncs.com"
      akId: ***
      akSecret: ***
      otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

Create OSS data volumes in Container Service console

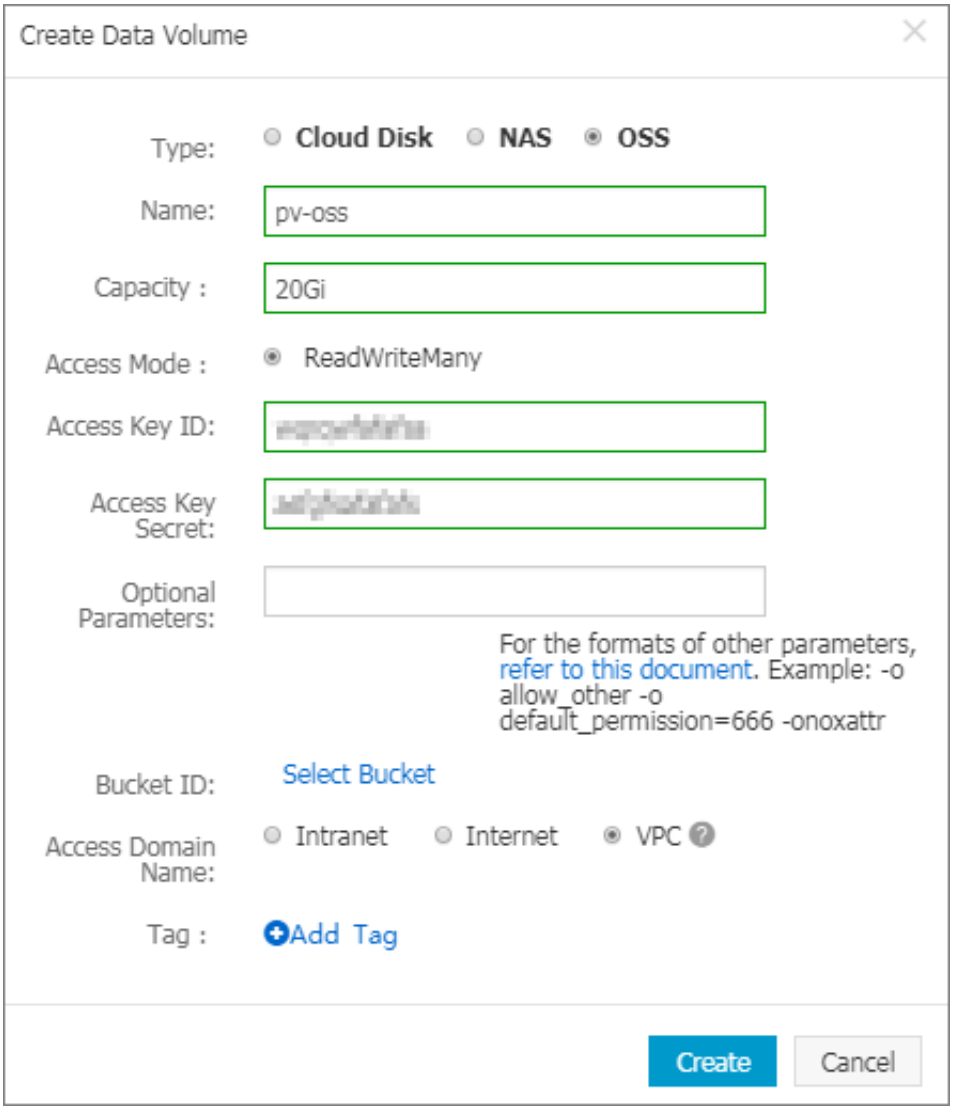
1. Log on to the *Container Service console*.
2. Click Kubernetes >**Clusters >** > **Storage**in the left-side navigation pane.
3. Select the cluster from the Clusters drop-down list and then click **Create** in the upper-right corner.



4. The Create Data Volume dialog box appears. Configure the data volume parameters.
   - **Type**: Select OSS in this example.
   - **Name**: Enter the name of the data volume you are about to create. The data volume name must be unique in the cluster. In this example, enter pv-oss.
   - **Capacity**: Enter the capacity of the data volume to be created.
   - **Access Mode**: ReadWriteMany by default.
   - **Access Key ID/Access Key Secret**: The AccessKey required to access OSS.
   - **Bucket ID**: Select the OSS bucket name you want to use. Click **Select Bucket**. Select the bucket in the displayed dialog box and click **Select**.
   - **Access Domain Name**: If the bucket and Elastic Compute Service (ECS) instance are in different regions, select **Internet**. If the bucket and ECS instance are in the same region,

select Intranet or VPC according to the cluster network type. Select **VPC** if the network type is Virtual Private Cloud (VPC) or select **Intranet** if the network type is classic network.

- **Tag**: Click Add Tag to add tags for this data volume.



5. Click **Create** after completing the configurations.

Step 2 Create PVC

Use the *oss-pvc.yaml* file to create the PVC.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-oss
spec:
  storageClassName: oss
  accessModes:
    - ReadWriteMany
  resources:
    requests:
```

```
      storage: 5Gi
```

Step 3 Create pod

Use the *oss-pod.yaml* file to create the pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-oss-example"
spec:
  containers:
    - name: "nginx"
      image: "nginx"
      volumeMounts:
          - name: pvc-oss
            mountPath: "/data"
  volumes:
  - name: pvc-oss
    persistentVolumeClaim:
        claimName: pvc-oss
```

Use OSS dynamic storage volumes

Currently not supported.

1.14 Storage claim management

1.14.1 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*.
- If you have already created a storage volume, use a cloud disk to create a cloud storage volume. For more information, see *Use Alibaba Cloud cloud disks*.

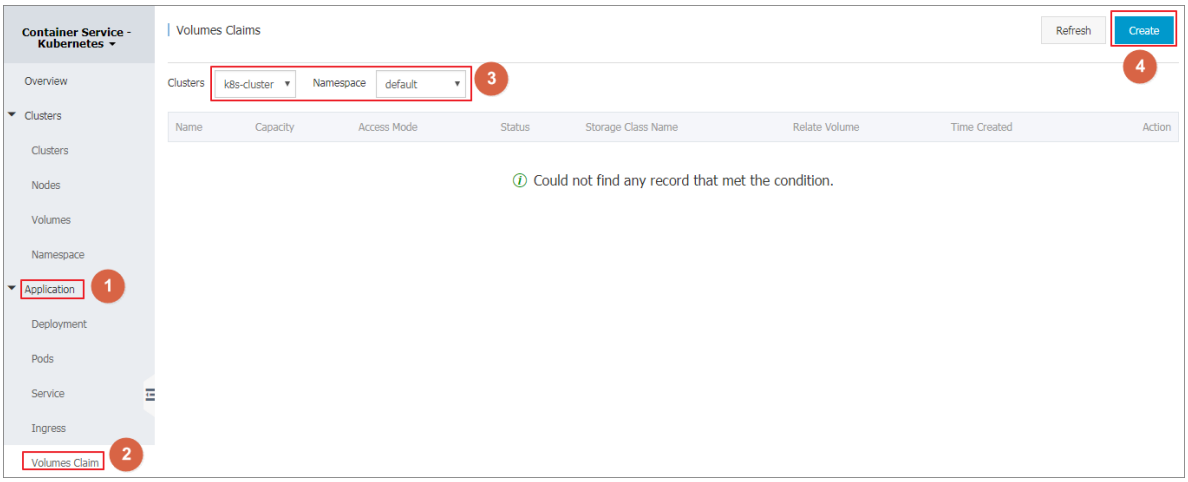
  By default, the storage claim is bound to the storage volume depending on the label **`alicloud-pvname`**. When the data volume is created by using the Container Service console, the storage volume is labeled by default. If the storage volume label does not exist, you must add a label before you select to bound this storage volume.

Context

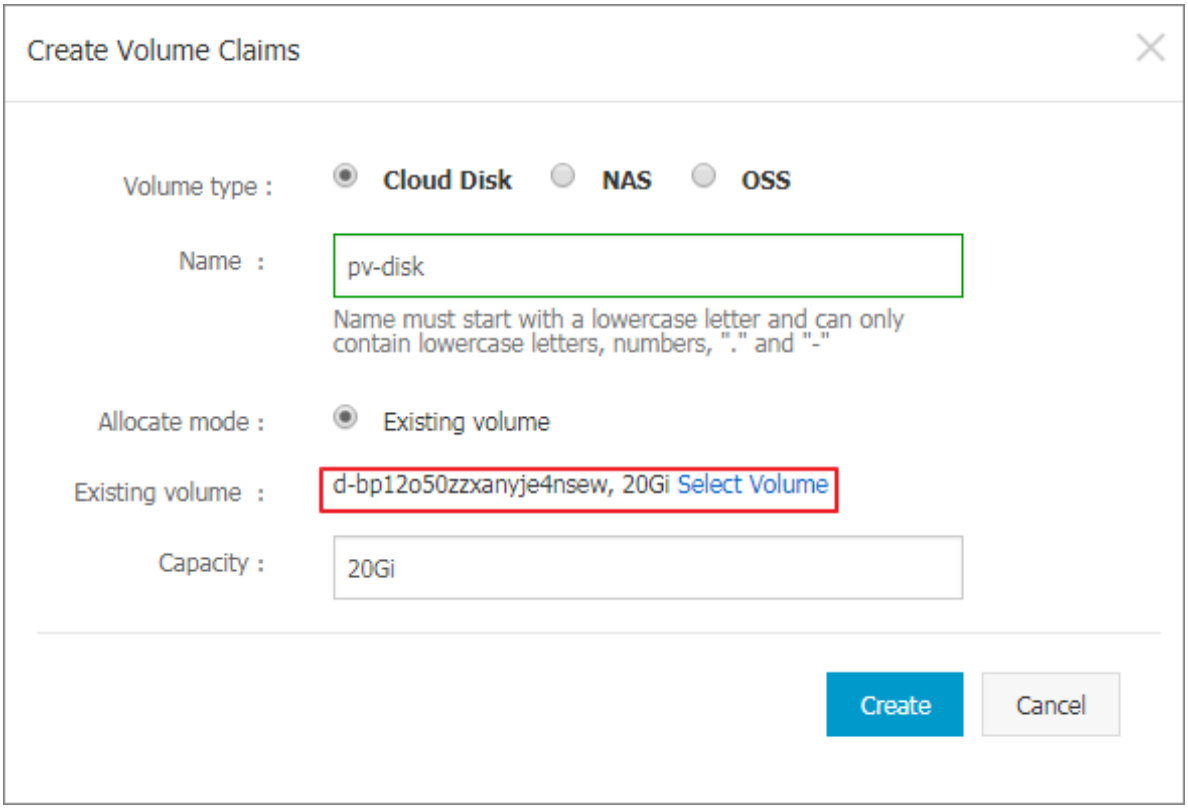
Procedure

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

2. Under Kubernetes, click **Application** > **Volumes Claim** in the left-side navigation pane to enter the Volumes Claims list page.
3. Select the target cluster and namespace, and click **Create** in the upper-right corner.



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

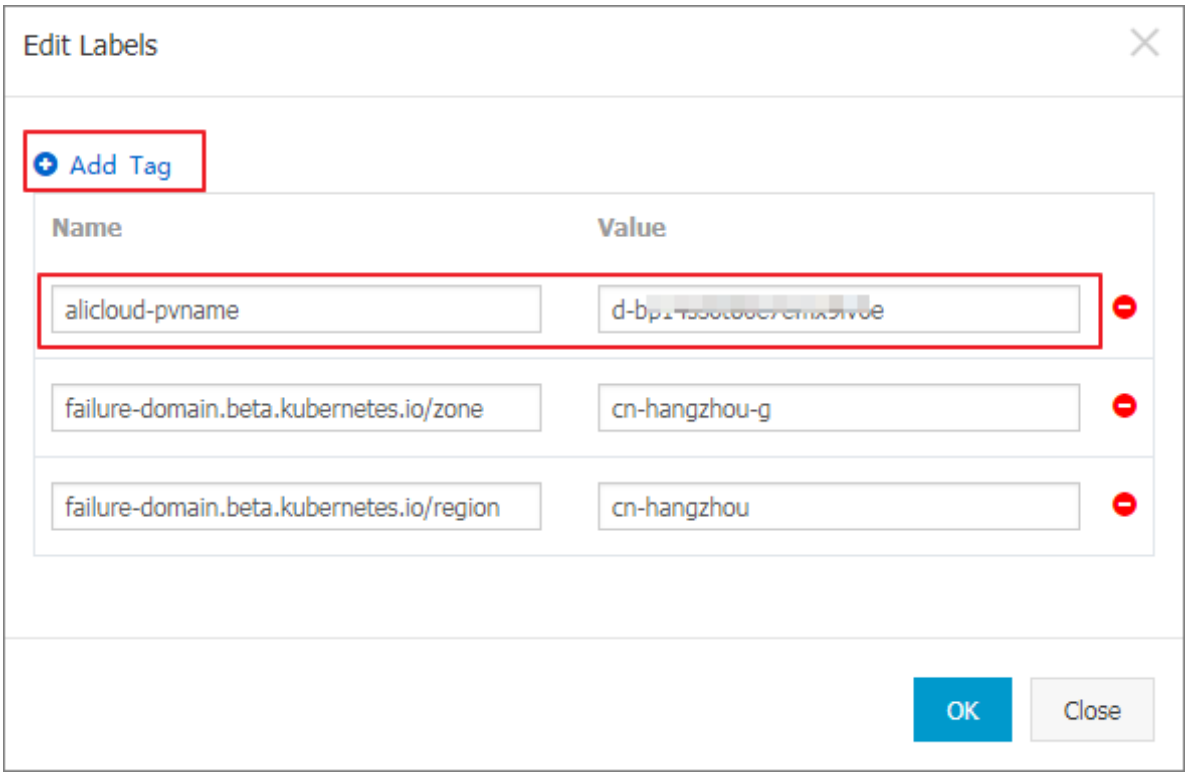


- **Volume claim type**: Consistent with storage volume, including cloud, NAS, and OSS types.
- **Name**: Enter the storage volume claim name.
- **Distribution mode**: Currently, only existing storage volumes are supported.
- **Existing storage volume**: Select to bound the storage volume of this type.

- **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.

1.14.2 Using persistent storage volume claim

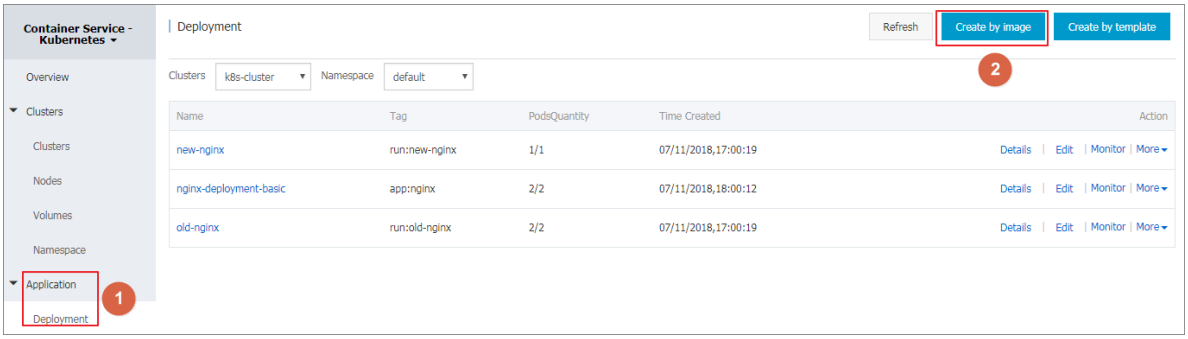
On the Container Service console, use an image or a template to deploy an application, so that you can use a persistent storage volume claim. In this example, an image is used to create an application. If you want to use a persistent storage volume claim with the template, see *Use Alibaba Cloud cloud disks*.

Prerequisites

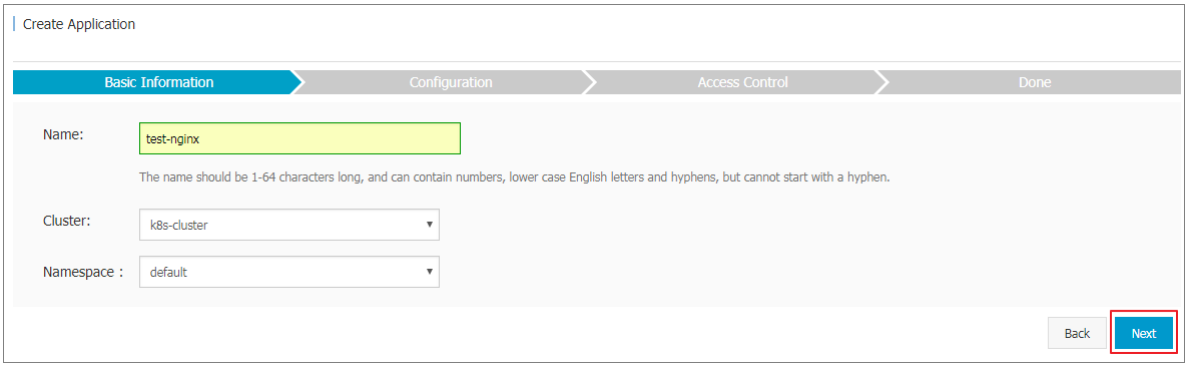
- You have created a Kubernetes cluster. For more information, see *Create a Kubernetes cluster*.
- If you have already created a storage volume claim, use the cloud disk to create a cloud disk storage volume claim PVC disk. For more information, see *Create a persistent storage volume claim*.

Procedure

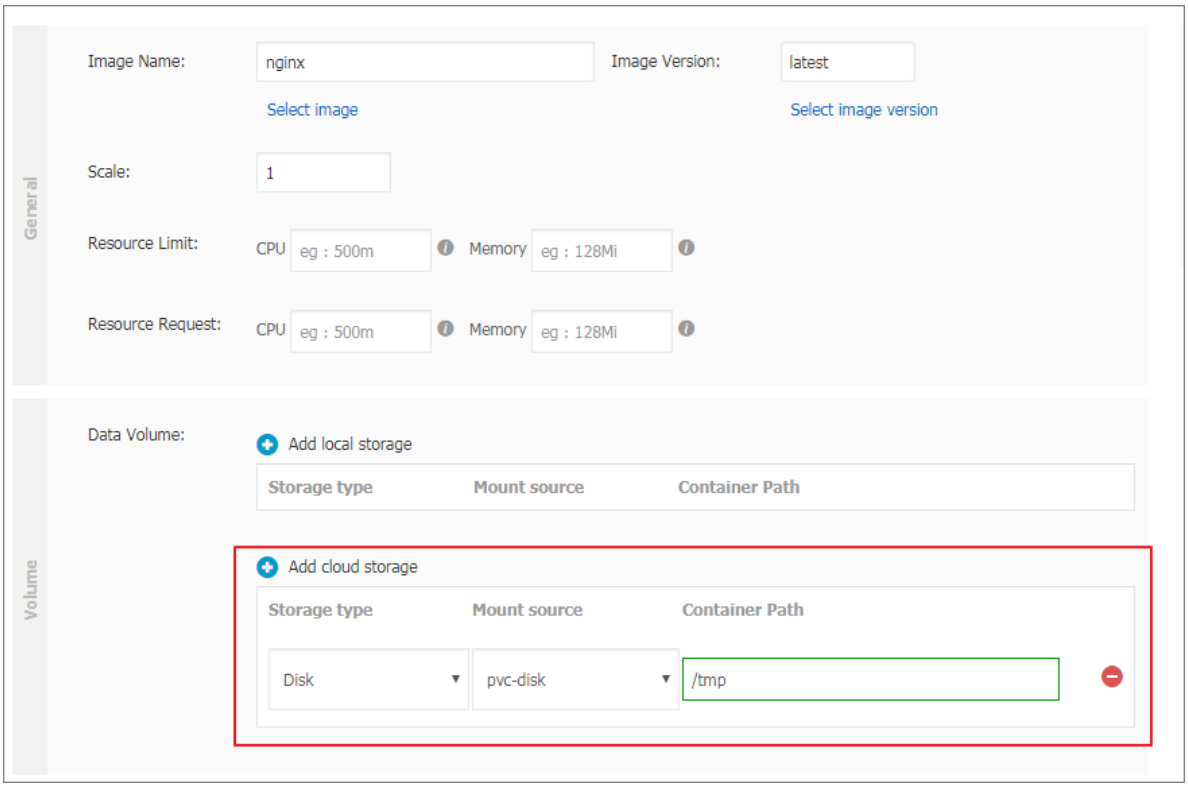
1. Log on to the *Container Service console*.
2. Under Kubernetes, click **Application** > **Deployment** in the left-side navigation pane. Enter the Deployment List page and click **Create by image** in the upper-right corner.



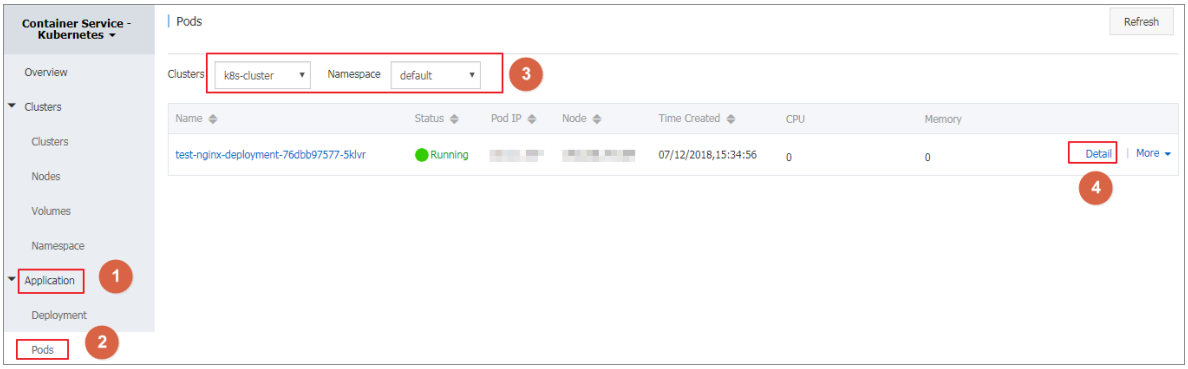
3. On the Basic Information page, configure the application name, deploy the cluster, and the namespace. Then click **Next**.



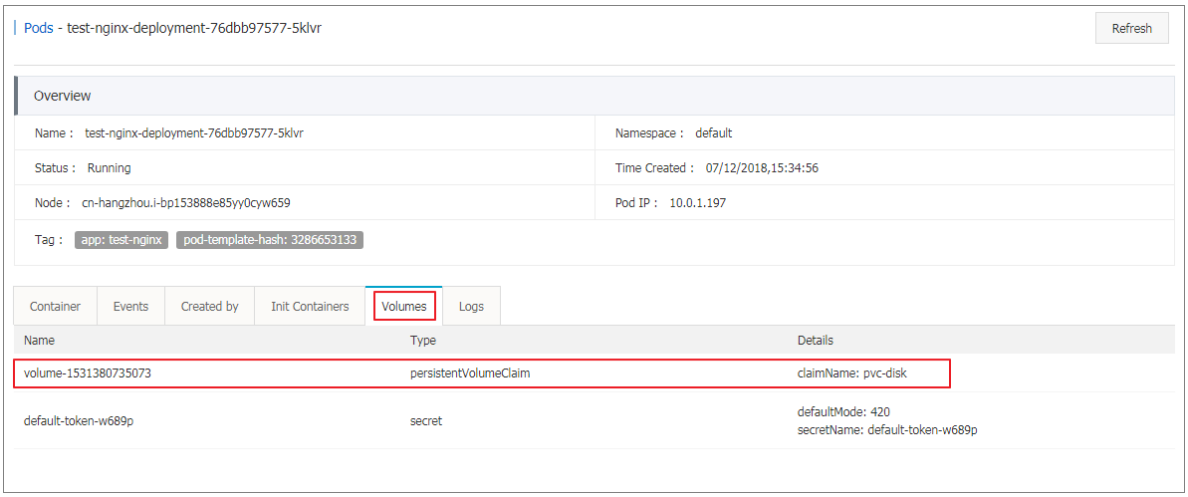
4. On the Application Configuration page, select Image. Then configure the cloud storage type of data volume, cloud disk, NAS, and OSS types are supported. In this example, use the cloud storage volume claim and click **Next**.



5. See *Create a service* 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.



1.15 Logs

1.15.1 Overview

The Alibaba Cloud Container Service Kubernetes cluster provides you with multiple methods to manage application logs.

- With the open-source Fluentd-pilot project provided by Alibaba Cloud Container Service, you can conveniently *#unique_88* to better use the features provided by Alibaba Cloud Log Service such as log statistics and analysis.
- *Configure Log4jAppender for Kubernetes and Log Service*.
- *A solution to log collection problems of Kubernetes clusters by using log-pilot, Elasticsearch, and Kibana*.

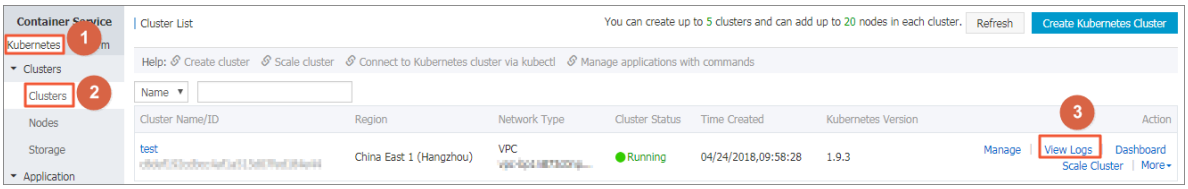
1.15.2 View cluster logs

Context

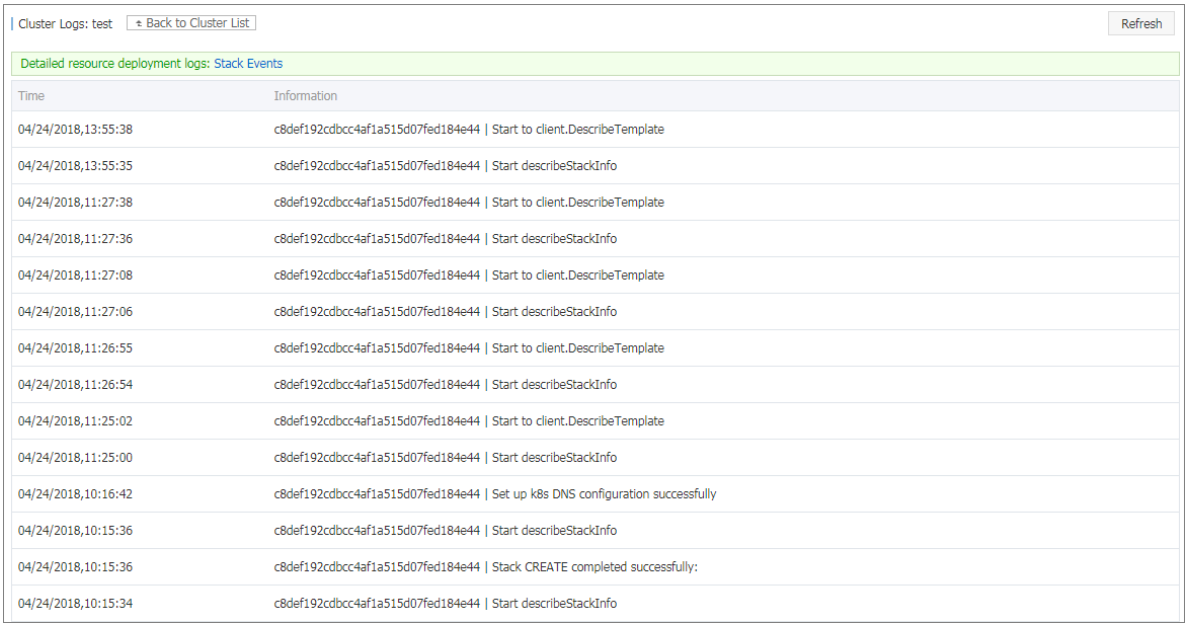
You can view the cluster operation logs by using the simple log service of Container Service.

Procedure

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



View the cluster operation information.



1.15.3 Collect Kubernetes logs

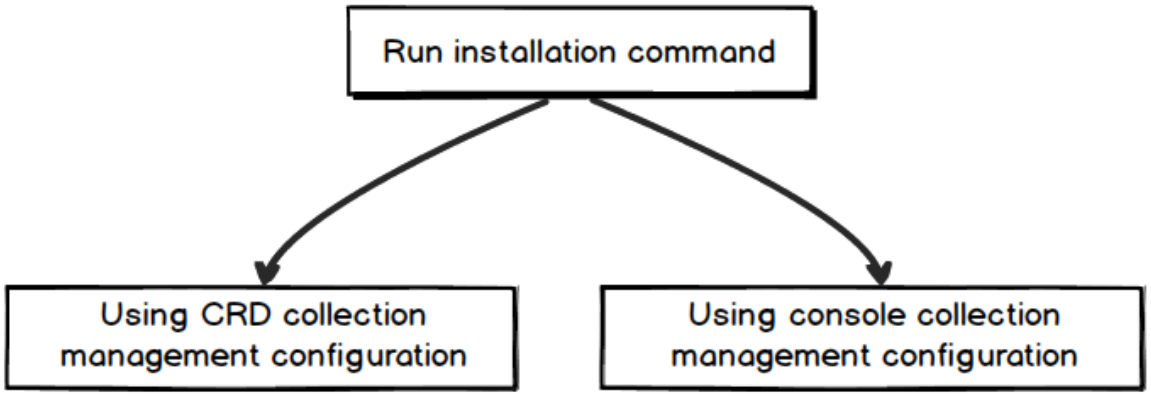
Log Service enables Logtail to collect Kubernetes cluster logs, and uses the CustomResourceDefinition (CRD) API to manage collection configurations. This document describes how to install and use Logtail to collect Kubernetes cluster logs.

Collection procedure

1. Install the alibaba-log-controller Helm package.
2. Configure the collection.

   You can configure the collection in the Log Service console or by using the CRD API as required. To configure the collection in the console, follow these steps:

Figure 1-1: Procedure



Step 1 Install the package.

1. Log on to the Master node of the Alibaba Cloud Container Service for Kubernetes.

   For how to log in, see *Access Kubernetes clusters by using SSH key pairs*.

2. Replace the parameters and run the following command.

   `${your_k8s_cluster_id}` to your Kubernetes cluster ID in the following installation command, and run this command:

```
 wget http://logtail-release.oss-cn-hangzhou.aliyuncs.com/linux64/
alicloud-log-k8s-install.sh -O alicloud-log-k8s-install.sh; chmod
744 ./alicloud-log-k8s-install.sh; sh ./alicloud-log-k8s-install.sh
 ${your_k8s_cluster_id}
```

Installation example

Run the installation command to obtain the following echo:

```
[root@iZbp******biaZ ~]# wget http://logtail-release.oss-cn-hangzhou.
aliyuncs.com/linux64/alicloud-log-k8s-install.sh -O alicloud-log-k8s-
install.sh; chmod 744 ./alicloud-log-k8s-install.sh; sh ./alicloud-log
-k8s-install.sh c12ba20**************86939f0b
....
....
....
alibaba-cloud-log/Chart.yaml
alibaba-cloud-log/templates/
alibaba-cloud-log/templates/_helpers.tpl
alibaba-cloud-log/templates/alicloud-log-crd.yaml
alibaba-cloud-log/templates/logtail-daemonset.yaml
alibaba-cloud-log/templates/NOTES.txt
alibaba-cloud-log/values.yaml
NAME: alibaba-log-controller
LAST DEPLOYED: Wed May 16 18:43:06 2018
NAMESPACE: default
```

```
STATUS: DEPLOYED


RESOURCES:
==> v1beta1/ClusterRoleBinding
NAME AGE
alibaba-log-controller 0s


==> v1beta1/DaemonSet
NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
logtail 2 2 0 2 0 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-ff6rf 0/1 ContainerCreating 0 0s
logtail-q5s87 0/1 ContainerCreating 0 0s
alibaba-log-controller-7cf6d7dbb5-qvn6w 0/1 ContainerCreating 0 0s


==> v1/ServiceAccount
NAME SECRETS AGE
alibaba-log-controller 1 0s


==> v1beta1/CustomResourceDefinition
NAME AGE
aliyunlogconfigs.log.alibabacloud.com 0s


==> v1beta1/ClusterRole
alibaba-log-controller 0s




[SUCCESS] install helm package : alibaba-log-controller success.
```

You can use helm status `helm status alibaba-log-controller` to check the current Pod status. The Running status indicates a successful installation.

Then, Log Service creates the project that is named starting with **k8s-log**. You can search for this project by using the **k8s-log** keyword in the Log Service console.

Step 2: Configure the collection.

To create Logstore and collect standard output (stdout) from all K8s containers, follow these steps:

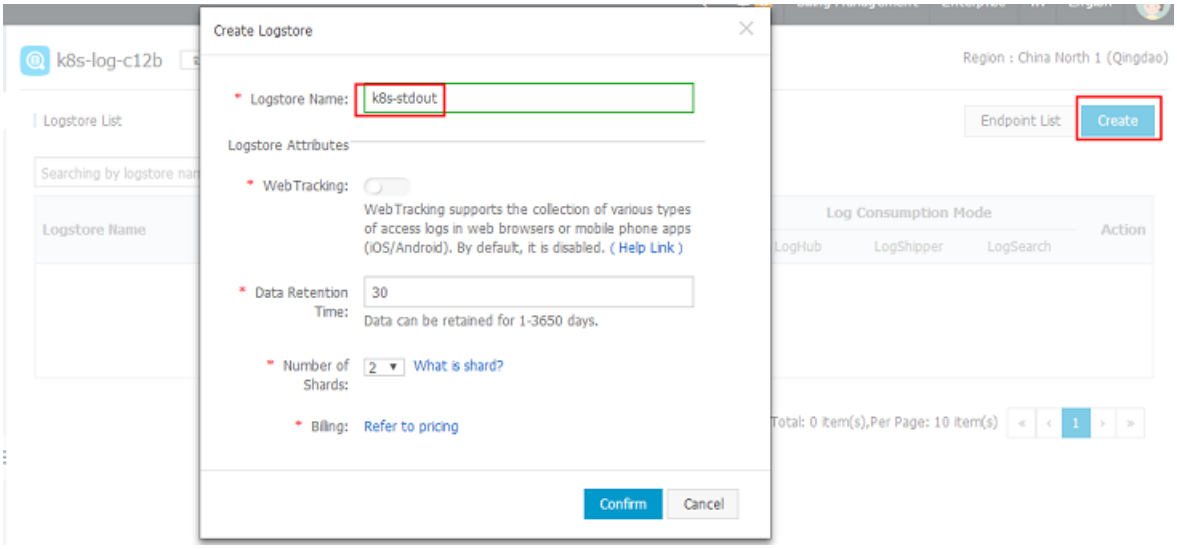
1. Go to the **Logstore List** page.

Click the project created in Step 1 to go to the Logstore List page.

2. Create Logstore.

   Click **Create** in the upper-right corner, and in the dialog box that appears, create Logstore.

   Figure 1-2: Creating Logstore

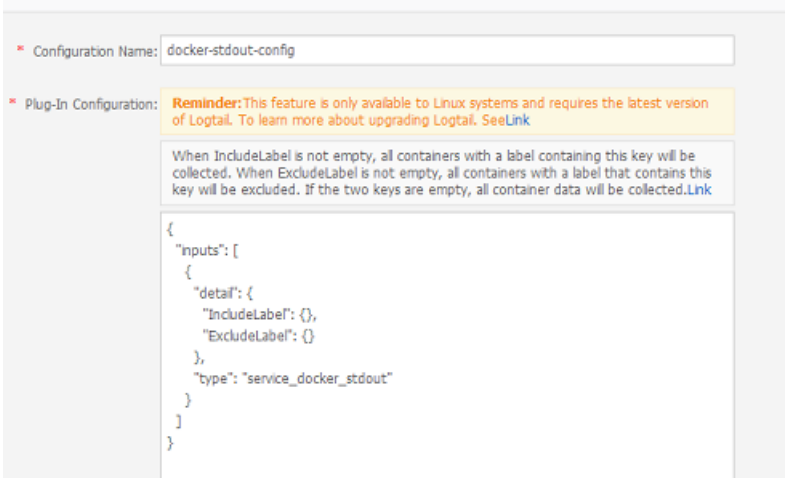


3. Configure the collection.

   1. Go to the **Data Import Wizard** page.
   2. Select **Docker Stdout** from **Third-Party Software**.

      Click **Apply to Machine Group** on the configuration pages. Then, you can collect all stdout files from all containers.

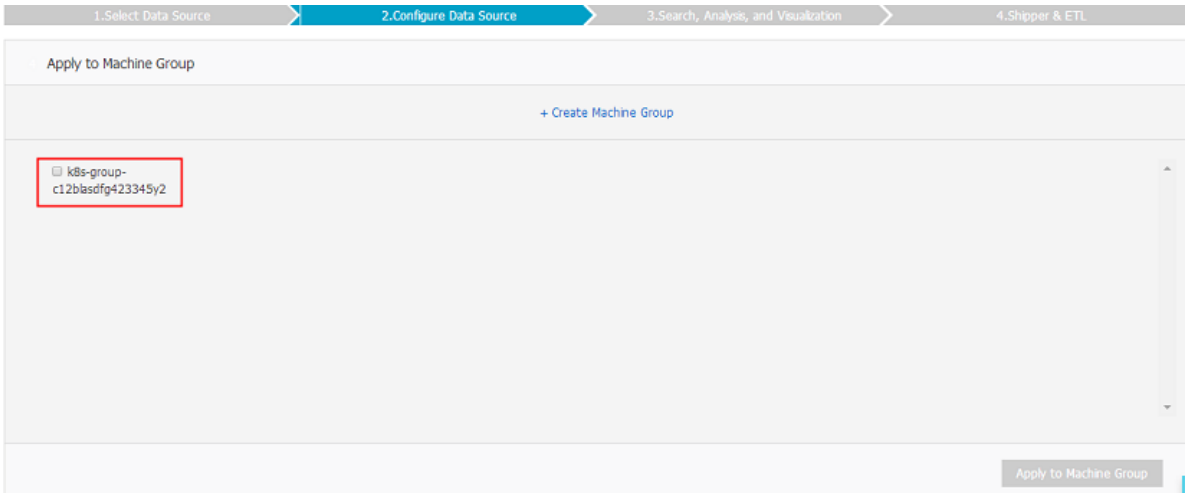
      Figure 1-3: Docker stdout



4. Apply the configuration to the machine group.

On the Apply to Machine Group page, select a machine group, and click Next.

Figure 1-4: Applying the configuration to the machine group

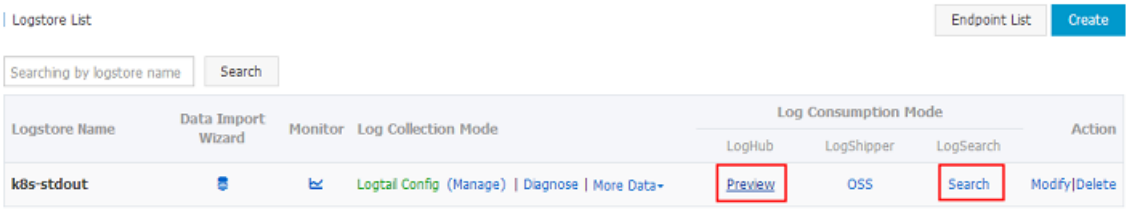


Now you have configured the collection. To configure indexes and log shipping, continue with the follow-up configurations. You can also exit the current page to complete the configuration.

View collected logs

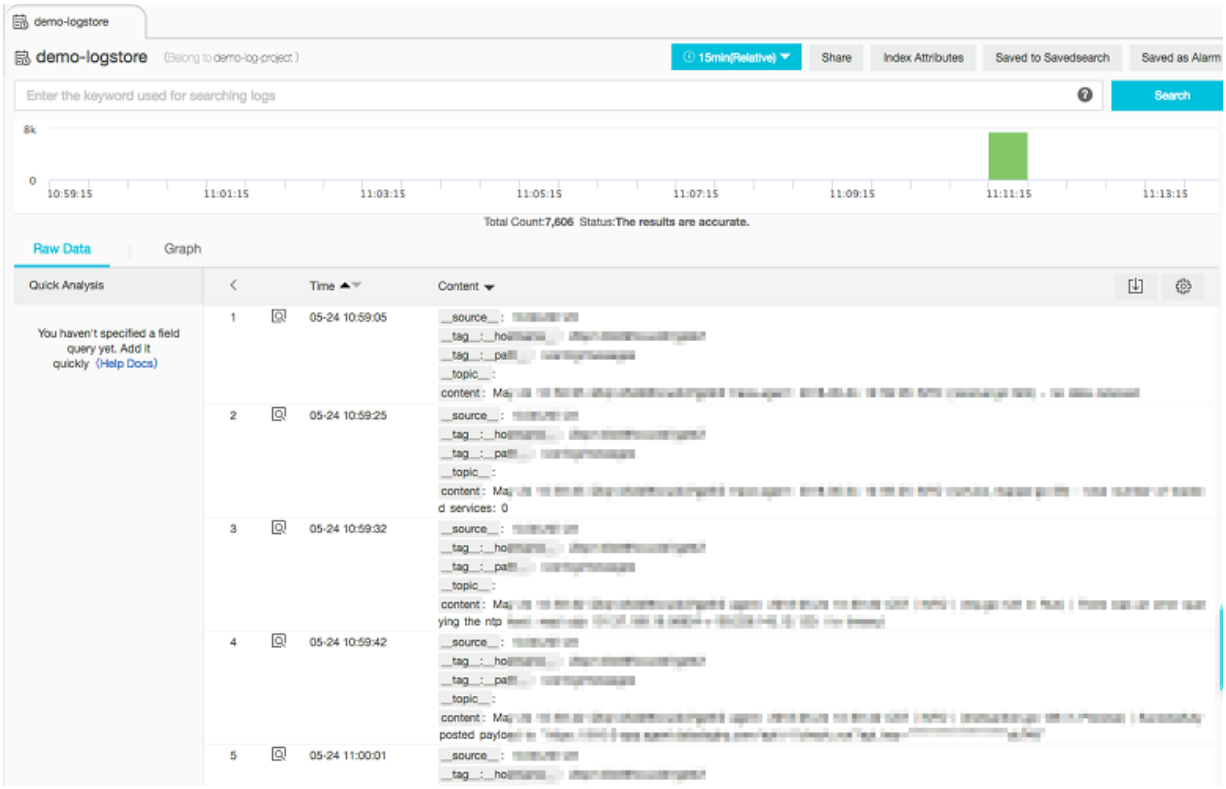
Based on the collection configuration, Logtail can collect stdout logs one minute after a container in your cluster receives stdout input. On the **Logstore List** page, click **Preview** to quickly preview collected logs, or click **Search** to customize searching and analysis of these logs.

Figure 1-5: Previewing and searching



As shown in the following image of the **Search** page, click any keyword of a log to start quick searching, or enter the keyword in the search box to search the specified logs.

Figure 1-6: Searching logs



Other methods for configuring collections

For more information about other methods for configuring collections, see:

Console Configuration

For more information about Console configuration, see:

- *Container text log (recommended)*
- *Container standard output (recommended)*
- *Host text file*

  By default, the root directory of the host is mounted to the `/logtail_host` directory of the Logtail container. You must add this prefix when configuring the path. For example, to collect data in the `/home/logs/app_log/` directory of the host, set the log path on the configuration page to `/logtail_host/home/logs/app_log/`.

CRD Configuration

For more information about CRD(CustomResourceDefinition) configuration, see *Configure Kubernetes log collection on CRD*.

1.15.4 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

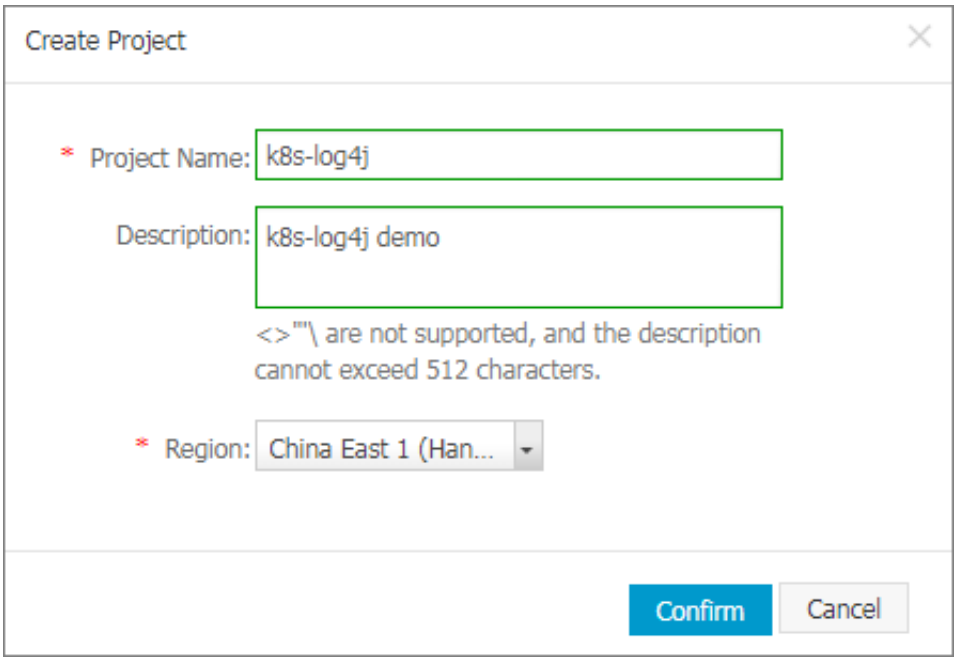
1. Log on to the *Log Service console*.
2. On the Project List page, click **Create Project** in the upper-right corner. Complete the configurations and then click **Confirm** to create the project.

   In this example, create a project named k8s-log4j and select the same region (China East 1 (Hangzhou)) as the Kubernetes cluster.

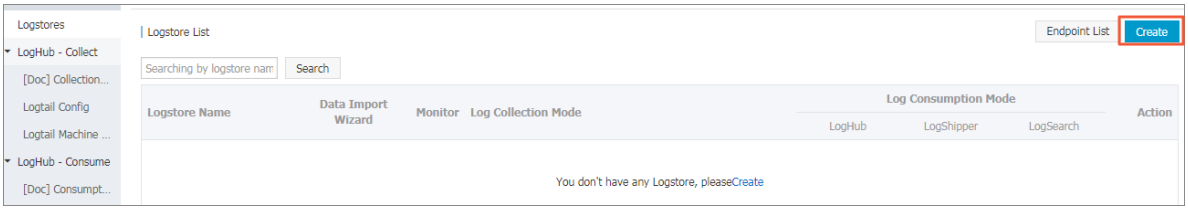
   

   Note:

   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.

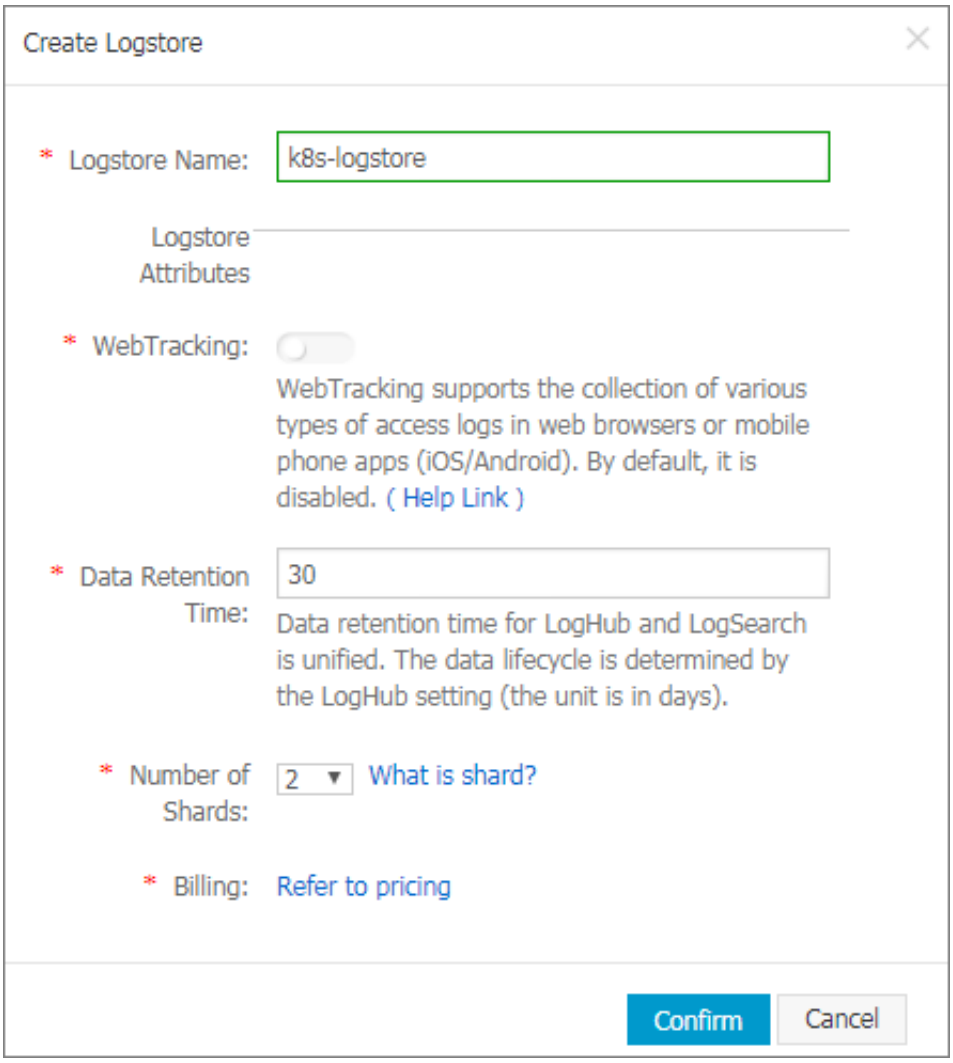


3. After being created, the project k8s-log4j is displayed on the Project List page. Click the project name.
4. The **Logstore List** page appears. Click **Create** in the upper-right corner.

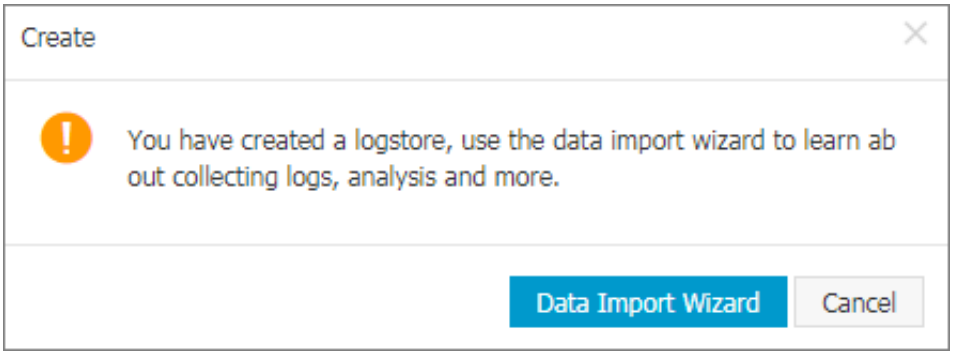


5. Complete the configurations and then click **Confirm**.

   In this example, create a Logstore named k8s-logstore.

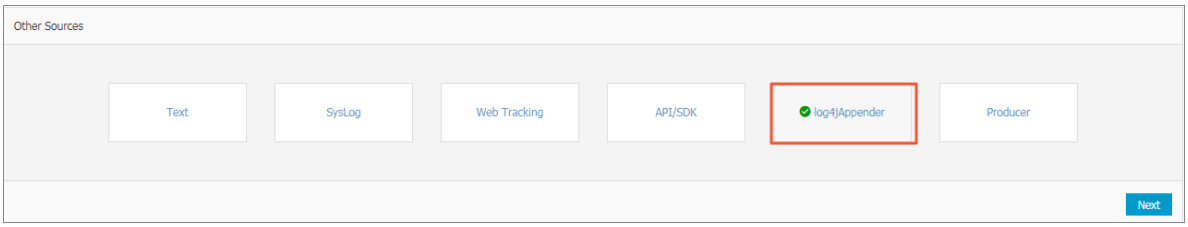


6. Then, a dialog box asking you to use the data import wizard appears.



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:

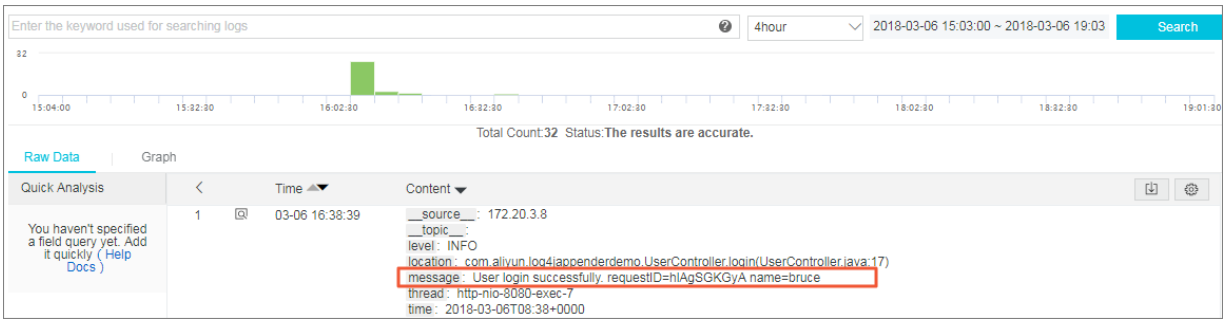
See *GitHub log4j-appender-demo* to view the complete collection of APIs.

```
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.

1.15.5 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"`.

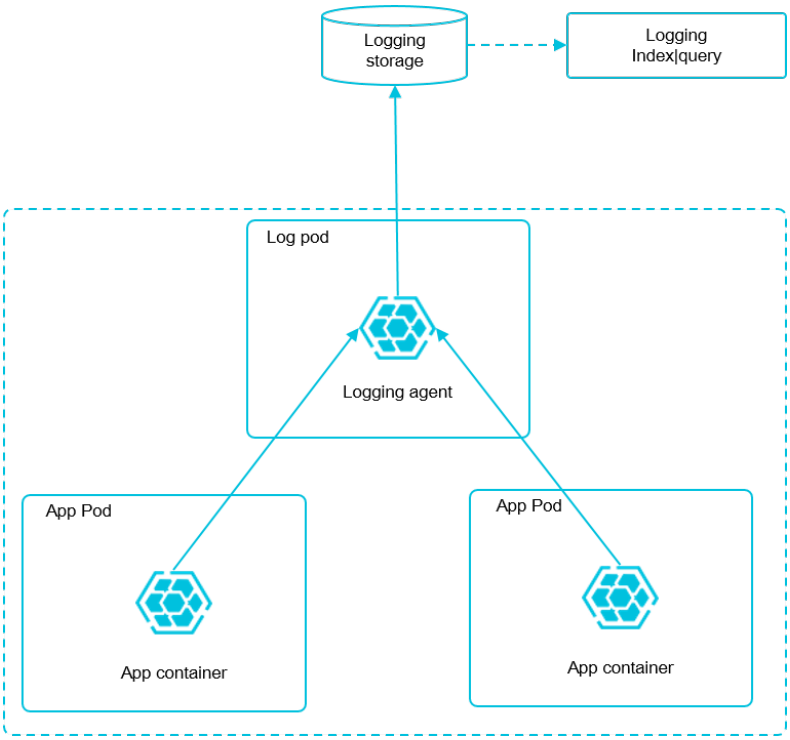
Besides, log-pilot supports multiple log parsing formats, including none, JSON, CSV, Nginx, apache2, and regxp. You can use the `aliyun_logs_$name_format=<format>` label to tell log-pilot to use what format to parse logs when collecting logs.

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 *Create a Kubernetes cluster* or *Access Kubernetes clusters by using SSH*.
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.

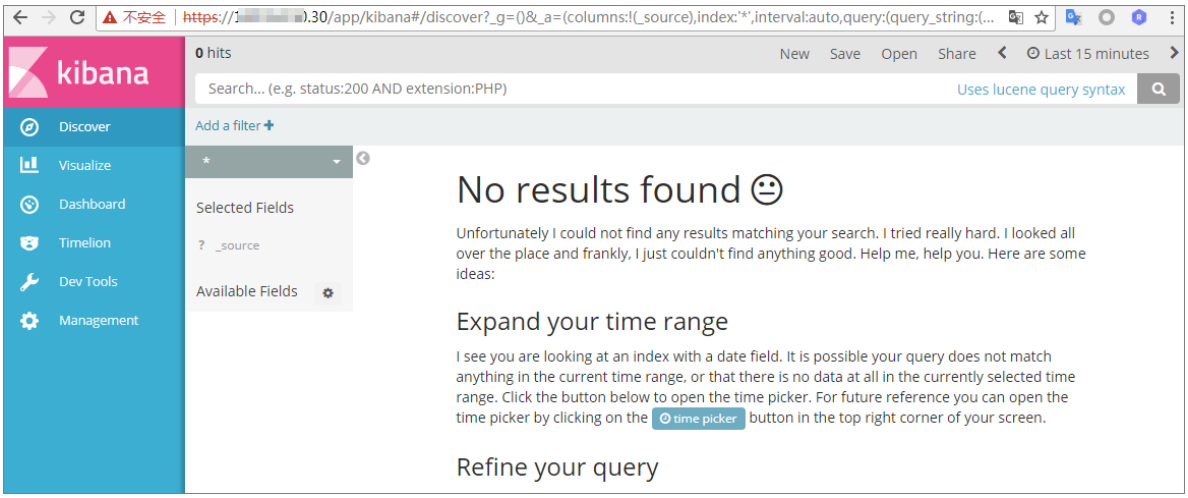
1. Create an Ingress to access the Kibana service from Internet. In this example, use the simple routing service to create an Ingress. For more information, see *Support for Ingress*. The orchestration template of the Ingress is as follows:

```
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.

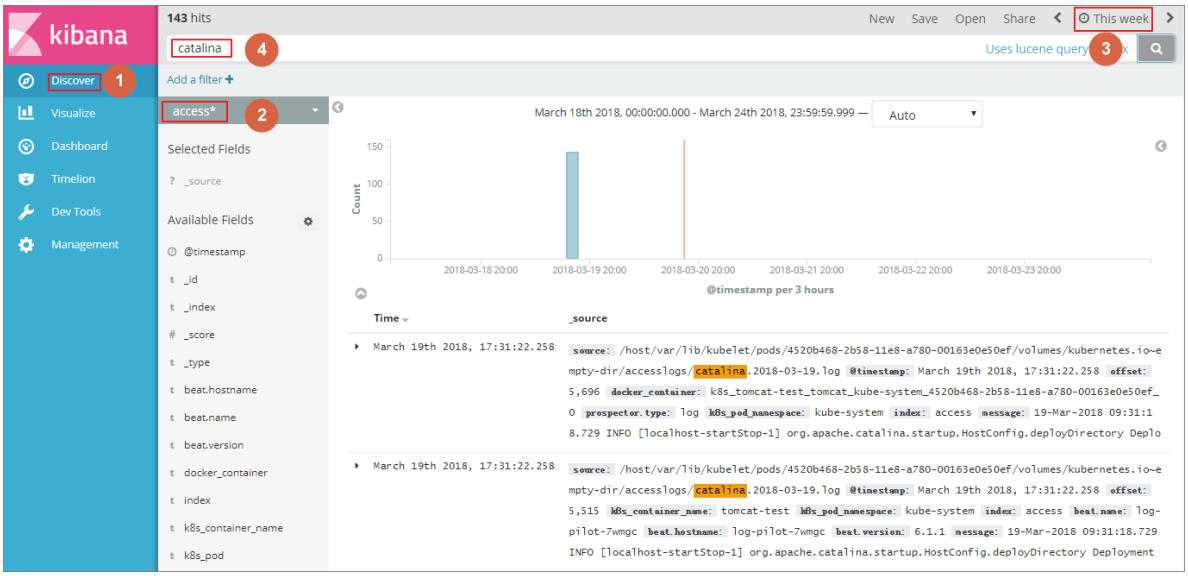


4. Click **Management** in the left-side navigation pane. Then, click **Index Patterns** > **Create Index Pattern**. The detailed index name is the `$name` variable suffixed with a time string. You can create an index pattern by using the wildcard `*`. In this example, use `$name*` to create an index 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.

1.16 Security

Authorization

The same as swarm clusters, Kubernetes clusters support authorizing RAM users to perform operations on clusters.

For more information, see *Use sub-accounts*.

Full-link TLS certificates

The following communication links in Container Service Kubernetes clusters are verified by TLS certificates to prevent the communication from being eavesdropped or tampered:

- `kubelet` on worker nodes actively communicates with `apiserver` on master nodes
- `apiserver` on master nodes actively communicates with `kubelet` on worker nodes

During initialization, the master node uses SSH tunnels to connect to the SSH service of other nodes (port 22) for initialization.

Native secret & RBAC support

Kubernetes secrets are used to store sensitive information such as passwords, OAuth tokens, and SSH keys. Using plain text to write sensitive information to a pod YAML file or a Docker image may leak the information, while using secrets avoids such security risks effectively.

For more information, see *Secret*.

Role-Based Access Control (RBAC) uses the Kubernetes built-in API group to drive authorization and authentication, which allows you to use APIs to manage pods that correspond to different roles, and the access permissions of roles.

For more information, see *Using RBAC authorization*.

Network policy

In a Kubernetes cluster, pods on different nodes can communicate with each other by default. In some scenarios, to reduce risks, the network intercommunication among different business services is not allowed and you must introduce the network policy. In Kubernetes clusters, you can use the Canal network driver to implement the support for network policy.

Image security scan

Kubernetes clusters can use Container Registry to manage images, which allows you to perform image security scan.

Image security scan identifies the security risks in images quickly and reduces the possibility of applications running on your Kubernetes cluster being attacked.

For more information, see *Image security scan*.

Security group and Internet access

By default, each newly created Kubernetes cluster is assigned a new security group with the minimal security risk. This security group only allows ICMP for the Internet inbound.

By default, you cannot use Internet SSH to access your clusters. To use Internet SSH to connect to the cluster nodes, see*Access Kubernetes clusters by using SSH* .

The cluster nodes access the Internet by using the NAT Gateway, which further reduces the security risks.

1.17 FAQ

1.17.1 Collect Kubernetes diagnosis information

1. Download diagnosis script on the master node and add the operation permission.

```
curl -o /usr/local/bin/diagnose_k8s.sh http://aliacs-k8s-cn-hangzhou
.oss-cn-hangzhou.aliyuncs.com/public/diagnose/diagnose_k8s.sh
 chmod u+x /usr/local/bin/diagnose_k8s.sh
```

2. Run the diagnosis script.

```
diagnose_k8s.sh

 + echo 'please get diagnose_1514939155.tar.gz for diagnostics' ##
The generated log file name is different every time you run the
diagnosis script.
 please get diagnose_1514939155.tar.gz for diagnostics
 + echo 'Upload diagnose_1514939155.tar.gz'
 Upload diagnose_1514939155.tar.gz
```

3. Upload the generated logs.

```
cd /usr/local/bin
 ls -ltr|grep diagnose_1514939155.tar.gz ##Replace with the
generated log file name.
```

1.17.2 FAQ about storage volumes

Storage volumes cannot be mounted

Check if flexvolume is installed.

Execute the following command on the master node:

```
# kubectl get pod -n kube-system | grep flexvolume

flexvolume-4wh8s 1/1 Running 0 8d
flexvolume-65z49 1/1 Running 0 8d
flexvolume-bpc6s 1/1 Running 0 8d
flexvolume-l8pml 1/1 Running 0 8d
flexvolume-mzkpv 1/1 Running 0 8d
flexvolume-wbfhv 1/1 Running 0 8d
flexvolume-xf5cs 1/1 Running 0 8d
```

Check if the flexvolume pod status is Running and if the number of running flexvolume pods is the same as the number of nodes.

If not, see *Install the plug-in*.

If the flexvolume pod status is not running, see the running log analysis of the plug-in.

Check if the dynamic storage plug-in is installed

To use the dynamic storage function of a cloud disk, execute the following command to verify the dynamic storage plug-in is installed:

```
# kubectl get pod -n kube-system | grep alicloud-disk

alicloud-disk-controller-8679c9fc76-lq6zb 1/1 Running 0 7d
```

If not, see *Install the plug-in*.

If the dynamic storage plug-in status is not running, see the running log analysis of the plug-in.

How to view types of storage logs?

View flexvolume logs by executing commands on the master1 node

Execute the following get command to view the error pod:

```
# kubectl get pod -n kube-system | grep flexvolume
```

Execute the following log command to view the log for the error pod:

```
# kubectl logs flexvolume-4wh8s -n kube-system
# kubectl describe pod flexvolume-4wh8s -n kube-system

# The last several lines in the pod description are the descriptio
ns of pod running status. You can analyze pod errors based on the
descriptions.
```

View drive logs of the cloud disk, Network Attached Storage (NAS), and Object Storage Service (OSS):

```
# View the persistent logs on the host node;
# If a pod mount fails, view the address of the node on which the pod
resides:

# kubectl describe pod nginx-97dc96f7b-xbx8t | grep Node
Node: cn-hangzhou.i-bp19myla3uvnt6zihejb/192.168.247.85
Node-Selectors: <none>

# Log on to the node to view logs:

# ssh 192.168.247.85
# ls /var/log/alicloud/flexvolume*
flexvolume_disk.log flexvolume_nas.log flexvolume_o#ss.log

You can see logs mounted on the cloud disk, NAS, and OSS;
```

View provsioner plug-in logs by executing commands on the master1 node

Execute the following get command to view the error pod:

```
# kubectl get pod -n kube-system | grep alicloud-disk
```

Execute the log command to view the log for the error pod:

```
# kubectl logs alicloud-disk-controller-8679c9fc76-lq6zb -n kube-
system
# kubectl describe pod alicloud-disk-controller-8679c9fc76-lq6zb -n
kube-system

# The last several lines in the pod description are the descriptio
ns of pod running status. You can analyze pod errors based on the
descriptions.
```

View Kubelet logs

```
# If a pod mount fails, view the address of the node on which the pod
resides:

# kubectl describe pod nginx-97dc96f7b-xbx8t | grep Node
Node: cn-hangzhou.i-bp19myla3uvnt6zihejb/192.168.247.85
Node-Selectors: <none>

# Log on to the node to view kubelet logs:

# ssh 192.168.247.85
# journalctl -u kubelet -r -n 1000 &> kubelet.log

# The value of -n indicates the number of log lines that you expect to
 see;
```

The above are methods to obtain error logs of flexvolume, provsioner, and kubelet. If the logs cannot help you to repair the status, contact Alibaba Cloud technical support with the logs.

FAQ about cloud disks

Cloud disk mount fails with timeout errors

If the node is added manually, the failure may be caused by problem about Security Token Service (STS) permissions. You need to manually configure Resource Access Management (RAM) permissions: *Use the instance RAM role in the console*.

Cloud disk mount fails with size errors

The following are size requirements for creating a cloud disk:

Note:

- Basic cloud disk: Minimum 5Gi
- Ultra cloud disk: Minimum 20Gi

- SSD cloud disk: Minimum 20Gi

Cloud disk mount fails with zone errors

When the ECS mounts a cloud disk, they must be in the same zone under the same region. Otherwise, the cloud disk cannot be mounted successfully.

After your system is upgraded, the cloud disk sometimes reports input/output error

1. Upgrade flexvolume to v1.9.7-42e8198 or later.
2. Rebuild pods that have already gone wrong.

Upgrading command:

```
# kubectl set image daemonset/flexvolume acs-flexvolume=registry.cn-
hangzhou.aliyuncs.com/acs/flexvolume:v1.9.7-42e8198 -n kube-system
```

Flexvolume version information: To obtain the latest version of flexvolume, log on to the container image service console, click **Image search** in the left-side navigation pane, and search for acs/flexvolume.

FAQ about NAS

NAS mount time is too long

If the NAS volume contains a large amount of files and the chmod parameter is configured in the mount template, the mount time may be too long. To solve this problem, remove the chmod parameter.

NAS mount fails with the timeout error

Check if the NAS mount point and the cluster are within the same Virtual Private Cloud (VPC). If not, NAS cannot be mounted.

FAQ about OSS

OSS mount fails

Check if the AK used is correct.

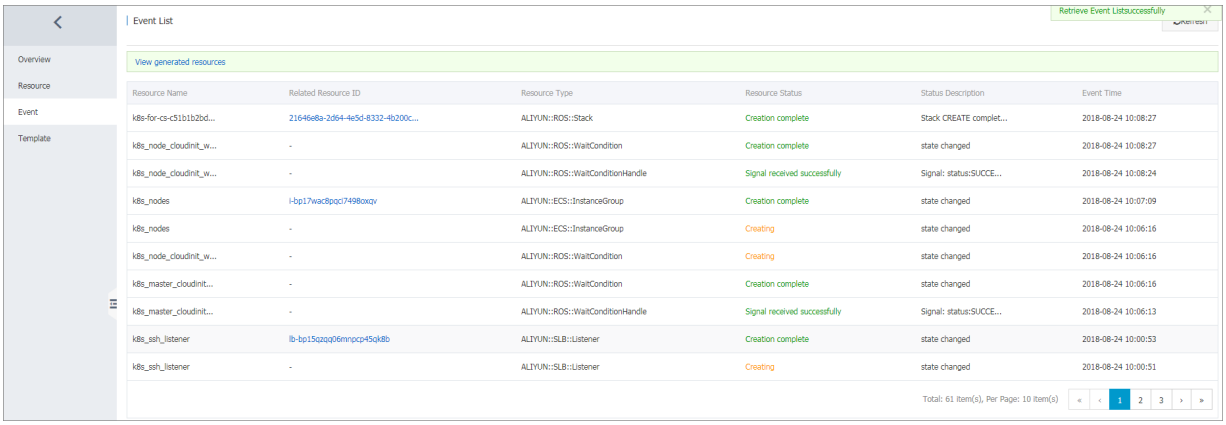
1.17.3 Failed to create a Kubernetes cluster

Check the cause of failure

You can check the cause of cluster creation failure by viewing the cluster creation events.

Log on to the *Resource Orchestration Service (ROS) console*.

Select the region in which the cluster resides. Click **Manage** at the right of the cluster. Click **Event** in the left-side navigation pane. Move the cursor over the failed event to view the specific error message of the failure.



If the preceding error message is displayed, it means that the cluster creation failed because the number of Virtual Private Cloud (VPC) instances has reached the quota.

Failure codes and solutions

- **Code: QuotaExceeded.Eip, Message: Elastic IP address quota exceeded**

  Solution: Release unused EIPs, or open a ticket to raise the EIP quota.
- **The maximum number of SLB instances is exceeded. Code: ORDER.QUANTITY_INVALID**

  Solution: Release unused SLB instances, or open a ticket to raise the SLB quota.
- **Resource CREATE failed: ResponseException: resources.k8s_vpc: VPC quota exceeded . Code: QuotaExceeded.Vpc**

  Solution: Release unused VPCs, or open a ticket to raise the VPC quota.
- **Resource CREATE failed: ResponseException: resources.k8s_master_1: The specified image does not support cloud-init. Code: ImageNotSupportCloudInit**

  Solution: When using custom image to create a cluster, the custom image used must be developed based on the lastest Centos public cloud image.
- **Status Code: 403 Code: InvalidResourceType.NotSupported Message: This resource type is not supported;**

  Solution: ECS is out of stock or the type of ECS instances you selected are not supported.

1.17.4 How to use private images in Kubernetes clusters

```
kubectl create secret docker-registry regsecret --docker-server=
registry-internal.cn-hangzhou.aliyuncs.com --docker-username=abc@
aliyun.com --docker-password=xxxxxx --docker-email=abc@aliyun.com
```

Where:

- **regsecret**: Specifies the secret key name and the name is customizable.
- **—docker-server**: Specifies the Docker repository address.
- **—docker-username**: Specifies the user name of the Docker repository.
- **—docker-password**: Specifies the logon password of the Docker repository.
- **—docker-email**: Specifies the email address (optional).

Add secret key parameters in the YML file.

```
containers:
    - name: foo
      image: registry-internal.cn-hangzhou.aliyuncs.com/abc/test:1.0
imagePullSecrets:
    - name: regsecret
```

Where:

- `imagePullSecrets` declares that a secret key must be specified when you pull the image.
- `regsecret` must be the same as the preceding secret key name.
- The Docker repository name in `image` must be the same as that in `--docker-server`.

For more information, see the official documentation *Use private repository*.

1.17.5 Upgrade Helm manually

Log on to the master node of the Kubernetes cluster, see *Connect to a Kubernetes cluster by using kubectl*.

Execute the following command:

```
helm init --tiller-image registry.cn-hangzhou.aliyuncs.com/acs/tiller:
v2.9.1 --upgrade
```

The image address can use the VPC domain name of the region corresponding to the image. For example, the image address of a machine in the Hangzhou region can be replaced by registry-vpc .cn-hangzhou.aliyuncs.com/acs/tiller:v2.9.1.

Wait for **`tiller`** passing through health check. Then you can execute `helm version` to view the upgraded version.

Note:

Only the Helm server version is upgraded here. To use the Helm client, download the corresponding client binary.

Helm 2.9.1 client download address: *https://github.com/kubernetes/helm/releases/tag/v2.9.1*。Currently, the latest version of Helm supported by Alibaba Cloud is 2. 9.1.

After the Helm client and server are both upgraded, you can see the following information by executing the **`helm version`** command:

```
#helm version
Client: &version.Version{SemVer:"v2.9.1", GitCommit:"a80231648a
1473929271764b920a8e346f6de844", GitTreeState:"clean" }
Server: &version.Version{SemVer:"v2.9.1", GitCommit:"a80231648a
1473929271764b920a8e346f6de844", GitTreeState:"clean "}
```