Alibaba Cloud Container Service

User Guide

Issue: 20190624

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

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

Contents

1 Kubernetes cluster

1.1 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 Clusters

1.2.1 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.

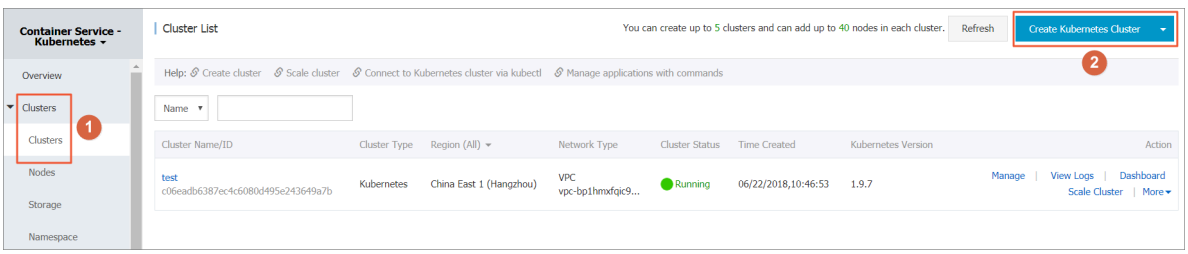
> **Note:**
> The deployment of Container Service Kubernetes clusters depends on the application deployment capabilities of Alibaba Cloud ROS. Therefore, activate ROS before creating a Kubernetes cluster.

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

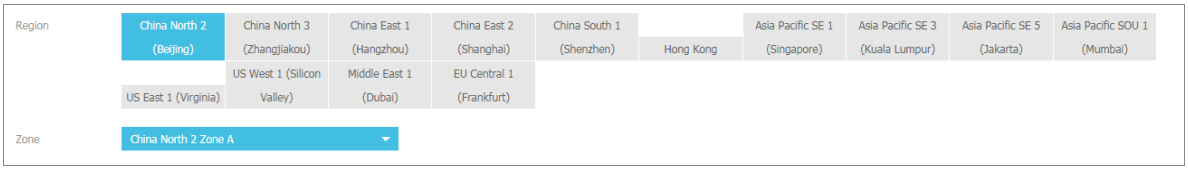
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Clusters in the left-side navigation pane to enter the Cluster List page.
3. Click Create Kubernetes Cluster in the upper-right corner.



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.

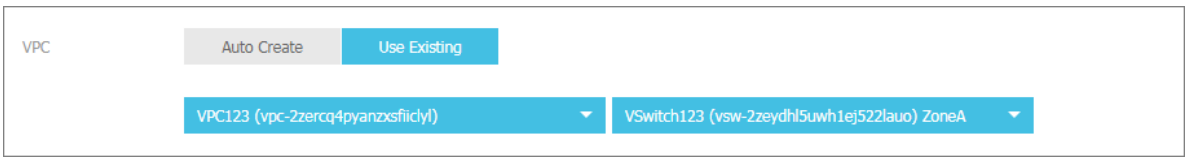


6. Set the cluster network type. Kubernetes clusters only support the VPC network type.

   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.

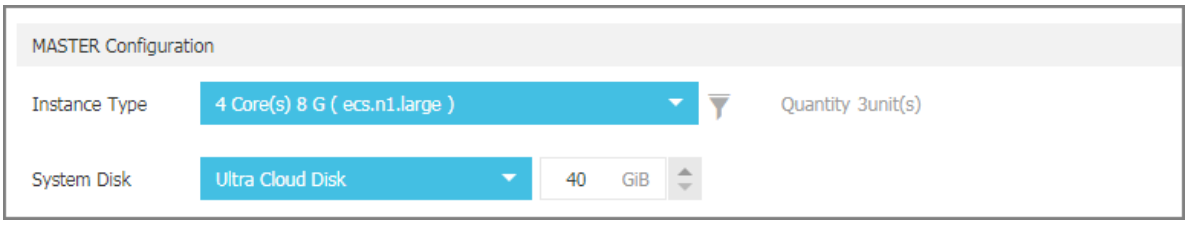


7. Configure the node type, Pay-As-You-Go and Subscription types are supported.

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.

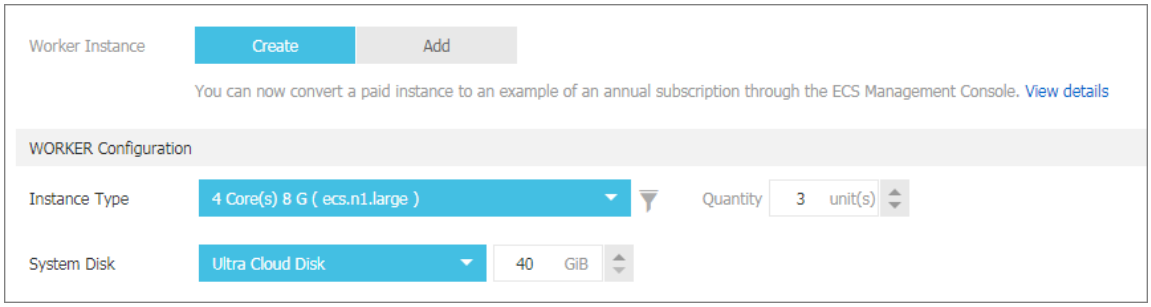
   

9. Configure the worker nodes. Select whether to create a worker node or add an existing ECS instance as the worker node.

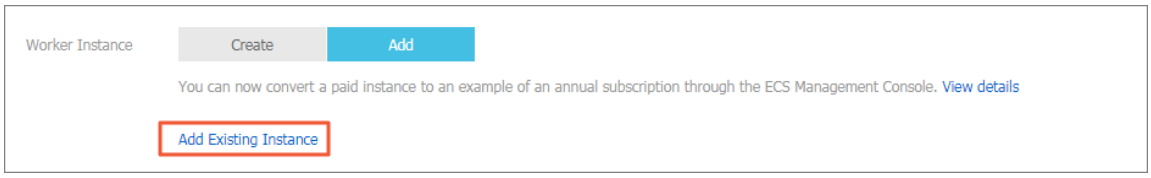
   > Note:
   >
   > · Currently, worker nodes only support the CentOS operating system.
   > · Each cluster can contain up to 37 worker nodes. To create more nodes, open a ticket.

- 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).



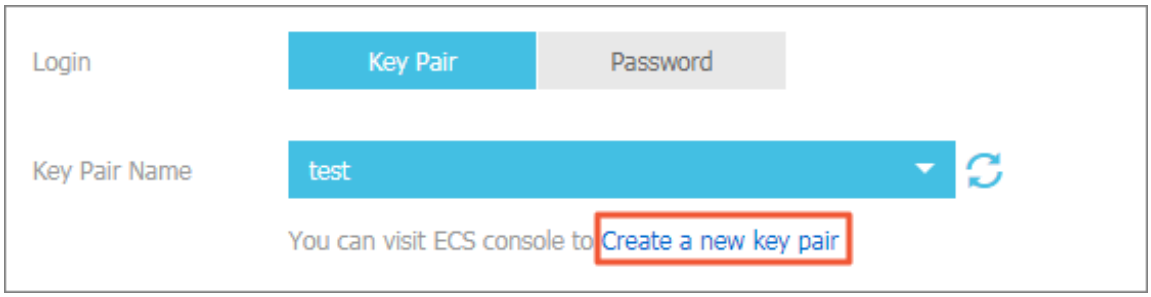
b. To add an existing ECS instance as the worker node, you must create an ECS instance in the current region in advance.



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.

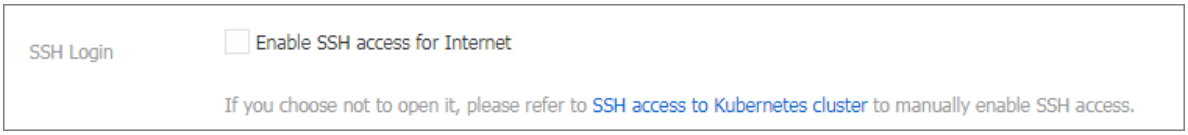
Specify the Pod Network CIDR and Service CIDR. Both of them cannot overlap with the Classless Inter-Domain Routing (CIDR) block used by VPC and the existing Kubernetes clusters in VPC, and you cannot modify the values after the cluster is created. Service address segment cannot be repeated with the Pod 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.

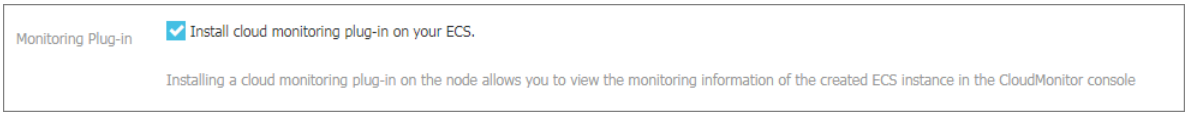
13. Select whether to enable SSH logon for Internet.

- 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 `NetworkPol icy` to define the

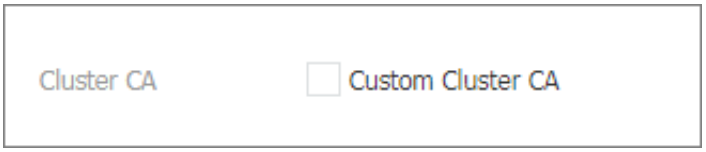
inter-container access policy. Supports bandwidth limiting for the separate containers. Currently it is in the public beta.

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.

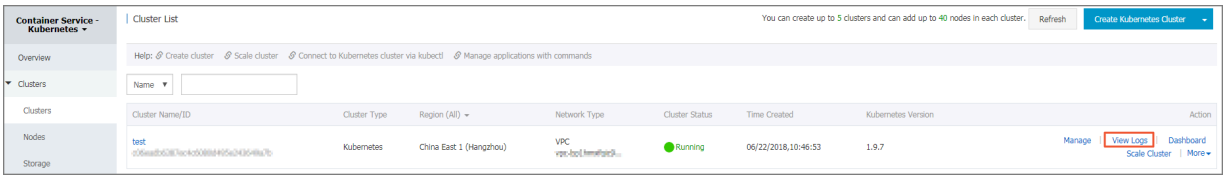


17.Click Create cluster to start the deployment.

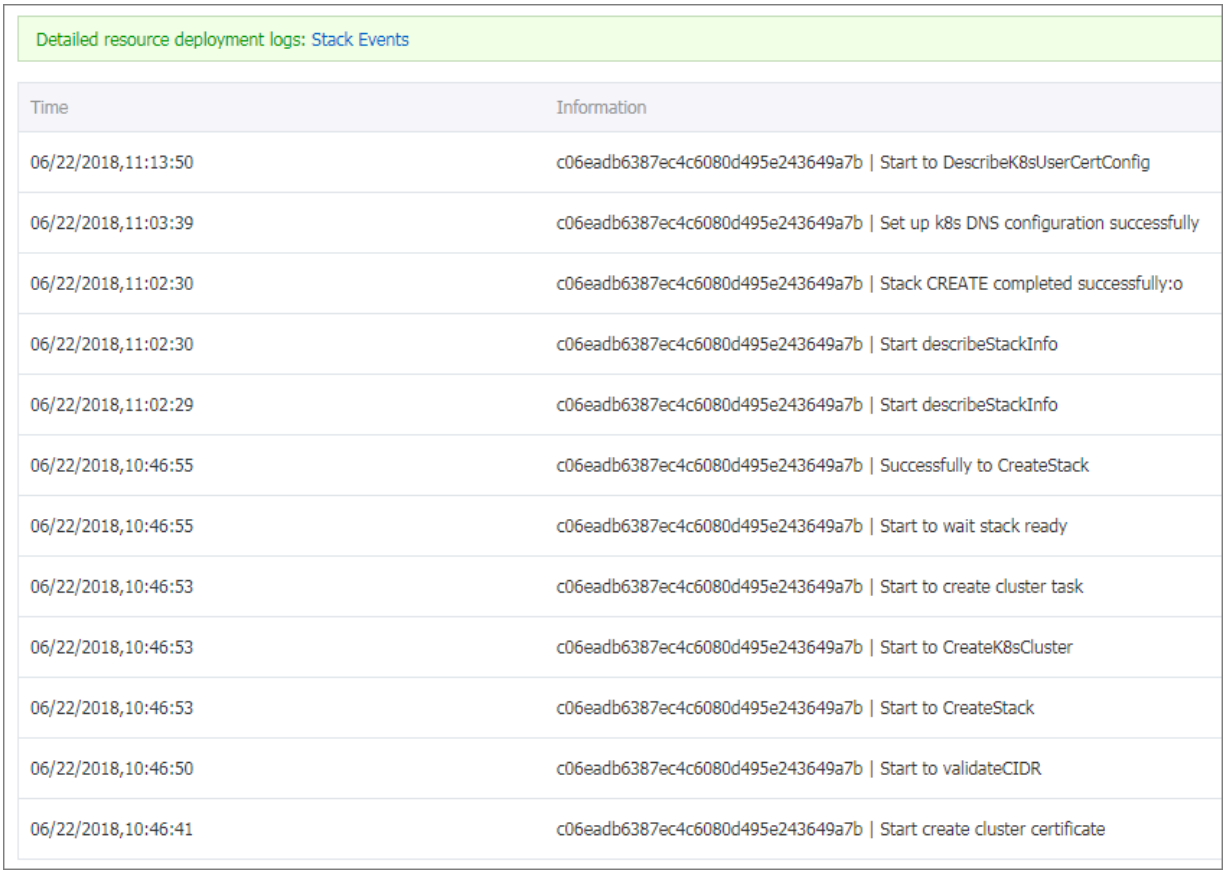
Note:
Creating a Kubernetes cluster with multiple nodes lasts more than 10 minutes.

Subsequent operations

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



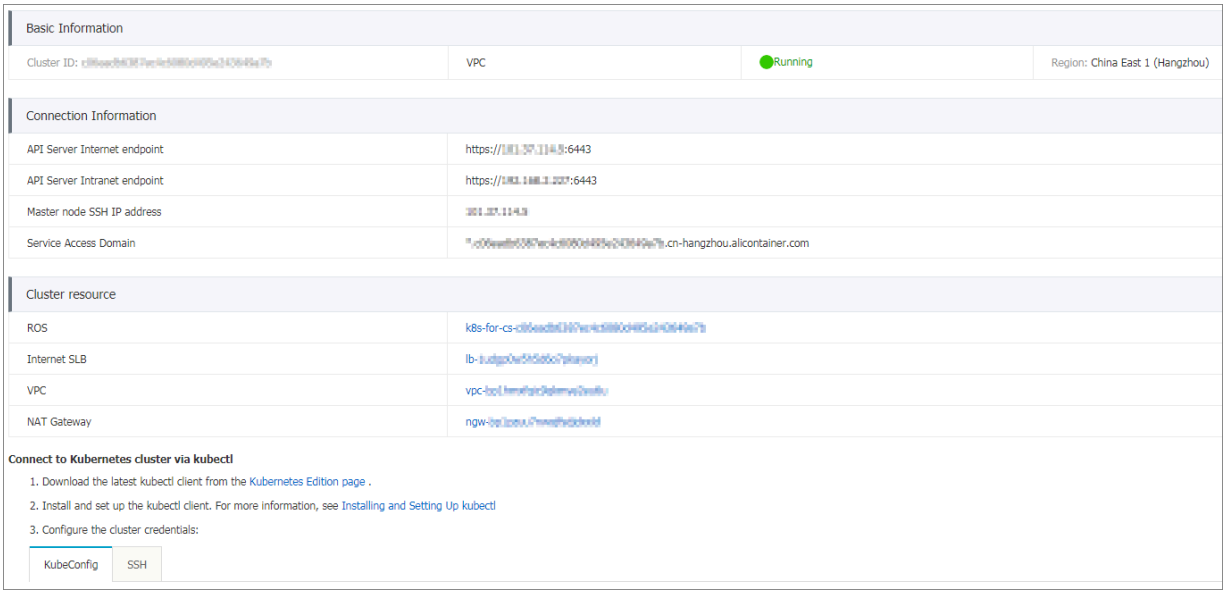
Click View Logs at the right of the cluster to view the cluster logs. To view more detailed information, click Stack Events.



Detailed resource deployment logs: Stack Events

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

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



Basic Information

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

Connection Information

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

Cluster resource

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

Connect to Kubernetes cluster via kubectl

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

KubeConfig SSH

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 >. alicontain er . 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.

1.2.3 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.

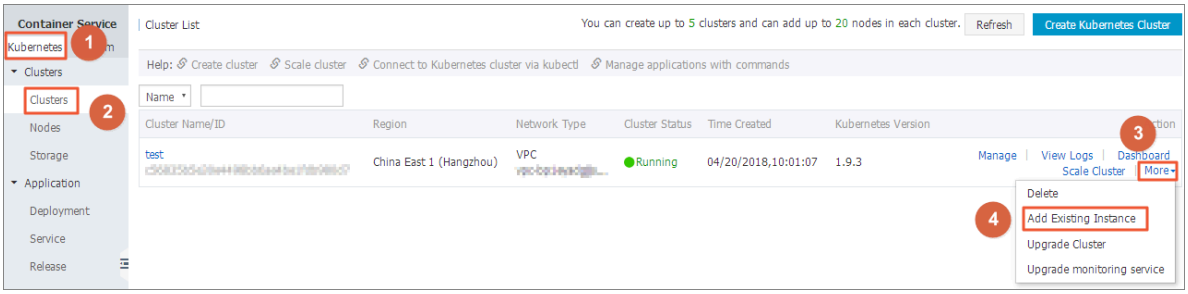
- The ECS instance to be added must be in the same Virtual Private Cloud (VPC) region as the cluster.
- When adding an existing instance, make sure that your instance has an Elastic IP (EIP) for the VPC network type, or the corresponding VPC is already configured with the NAT gateway. In short, make 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.

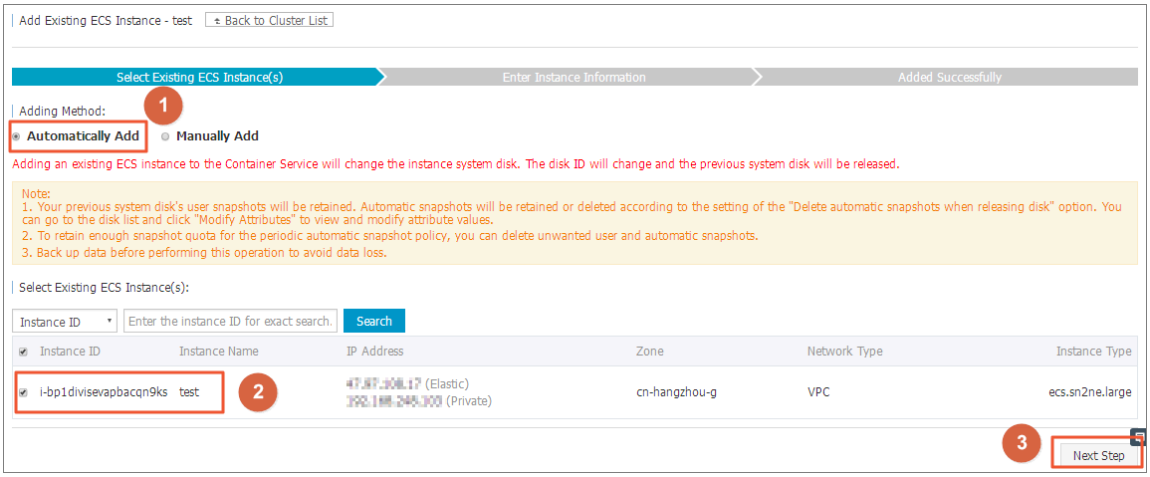
   On the Add Existing ECS Instance page and you can automatically or manually add an existing instance.

   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.



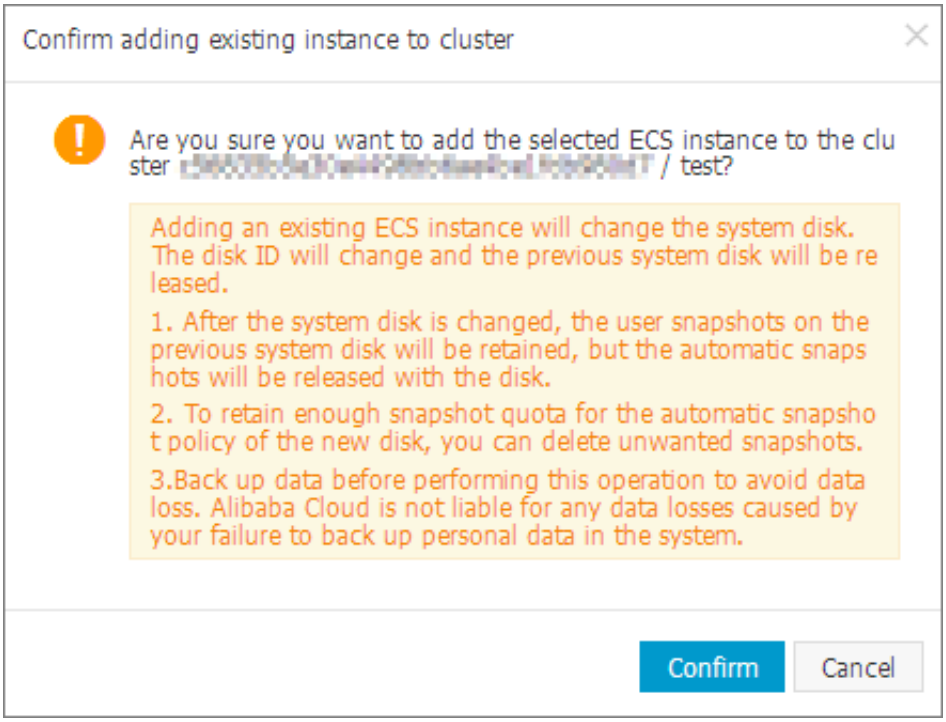
4. Select Automatically Add to add multiple ECS instances at a time.

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



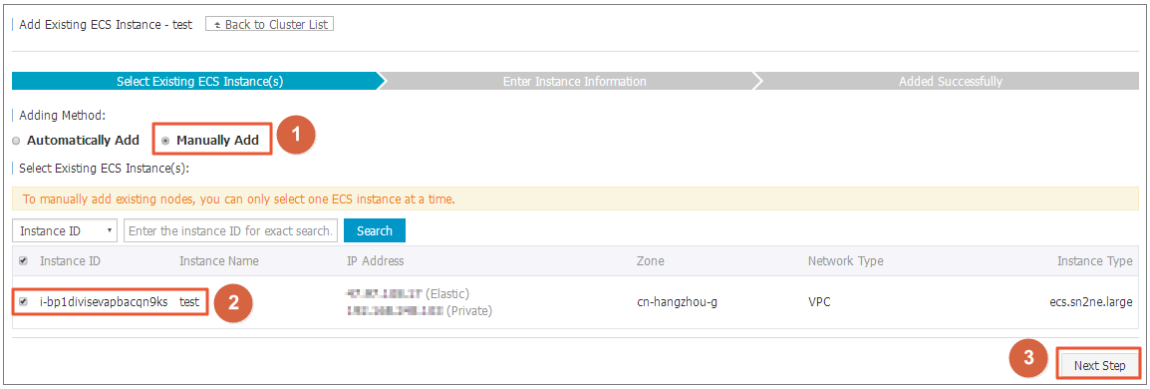
   b) Enter the instance information, set the logon password, and then click Next Step.

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

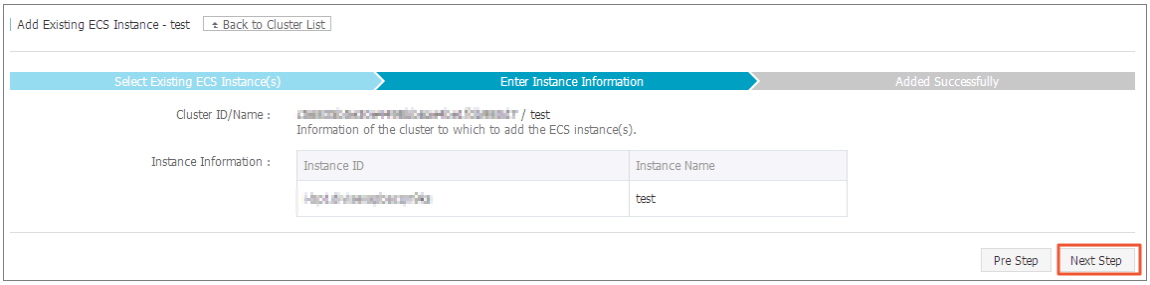

Confirm adding existing instance to cluster
Are you sure you want to add the selected ECS instance to the cluster / test?
Adding an existing ECS instance will change the system disk. The disk ID will change and the previous system disk will be released.
1. After the system disk is changed, the user snapshots on the previous system disk will be retained, but the automatic snapshots will be released with the disk.
2. To retain enough snapshot quota for the automatic snapshot policy of the new disk, you can delete unwanted snapshots.
3.Back up data before performing this operation to avoid data loss. Alibaba Cloud is not liable for any data losses caused by your failure to back up personal data in the system.
Confirm
Cancel

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

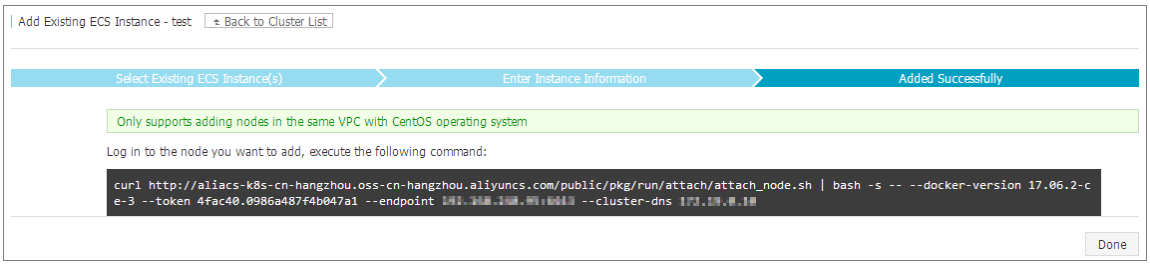
   a) Select the ECS instance to be added and then click Next Step. You can only add one ECS instance at a time.



   b) Confirm the information and then click Next Step.

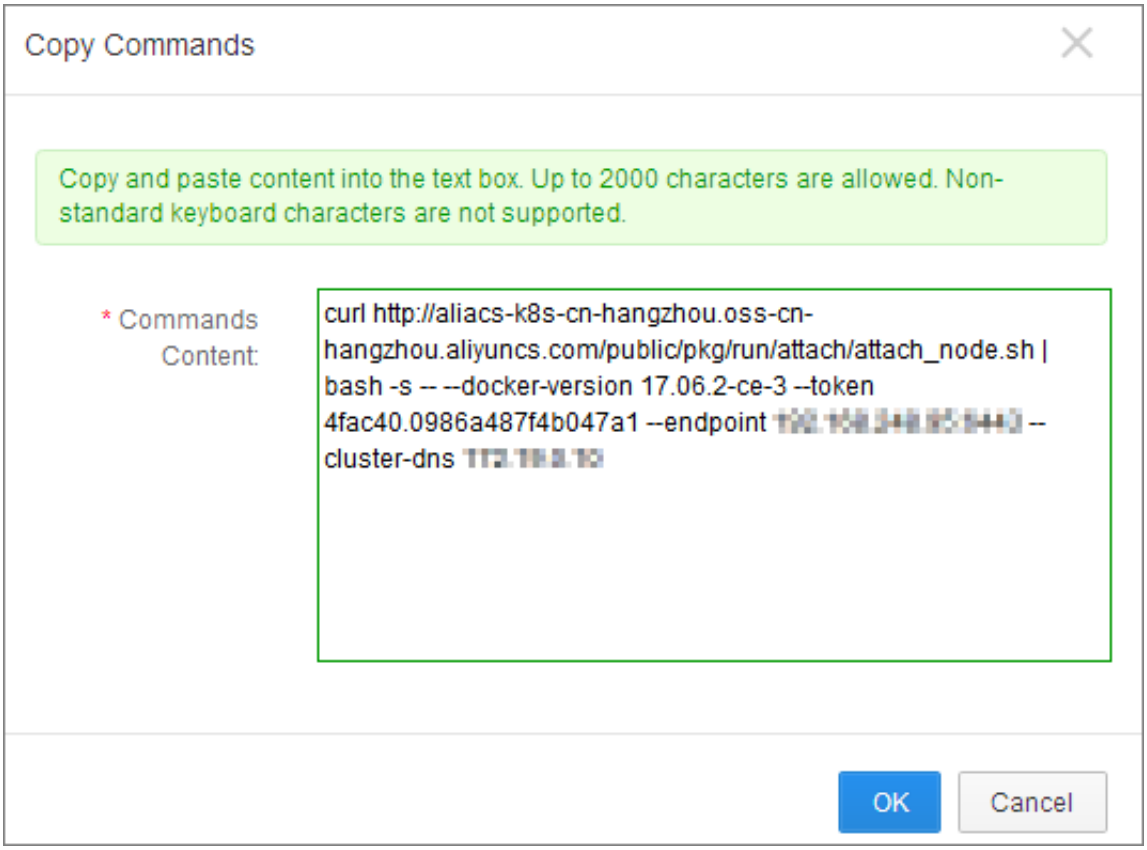


   c) Go to the Add Existing ECS Instance page and copy the command.



   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.

1.2.4 Scale out or in a cluster

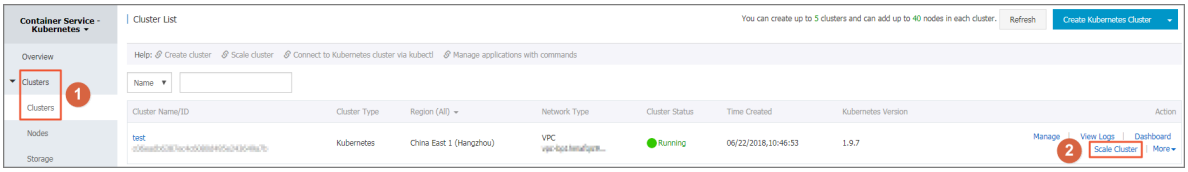
In the Container Service console, you can scale out or scale in the worker nodes of a Kubernetes cluster according to your actual business requirements.

Context

- Currently, Container Service does not support scaling in 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.

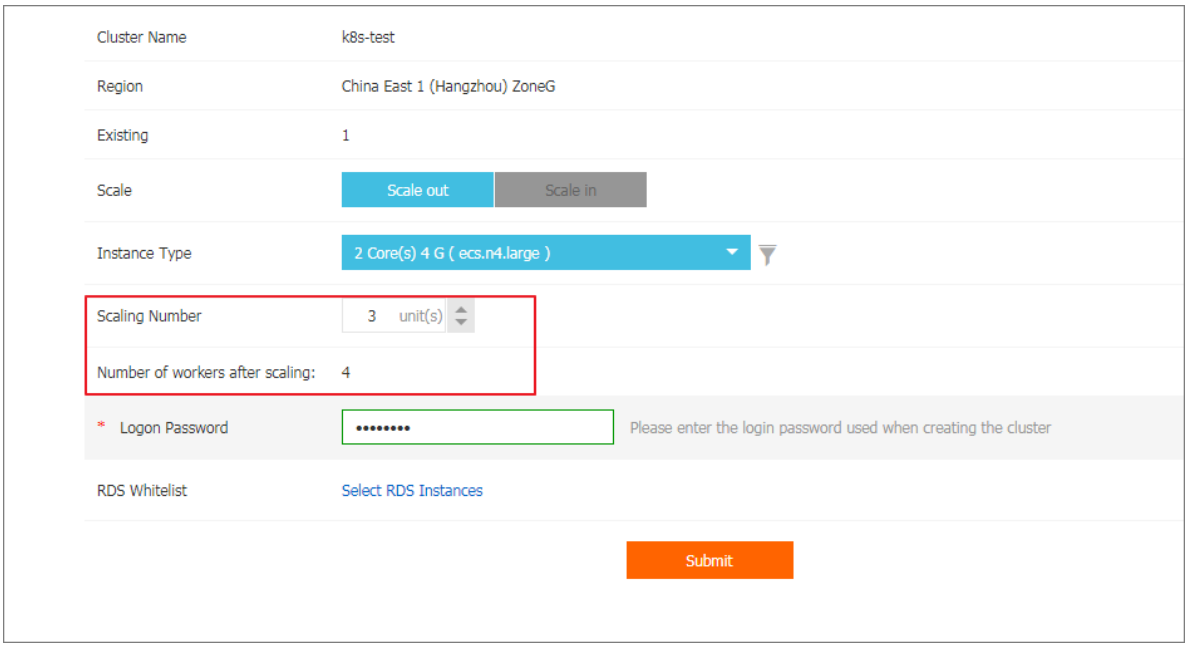
Procedure

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



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

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



5. Enter the logon password of the node.

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

6. Click Submit.

What's next

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

1.2.5 View cluster overview

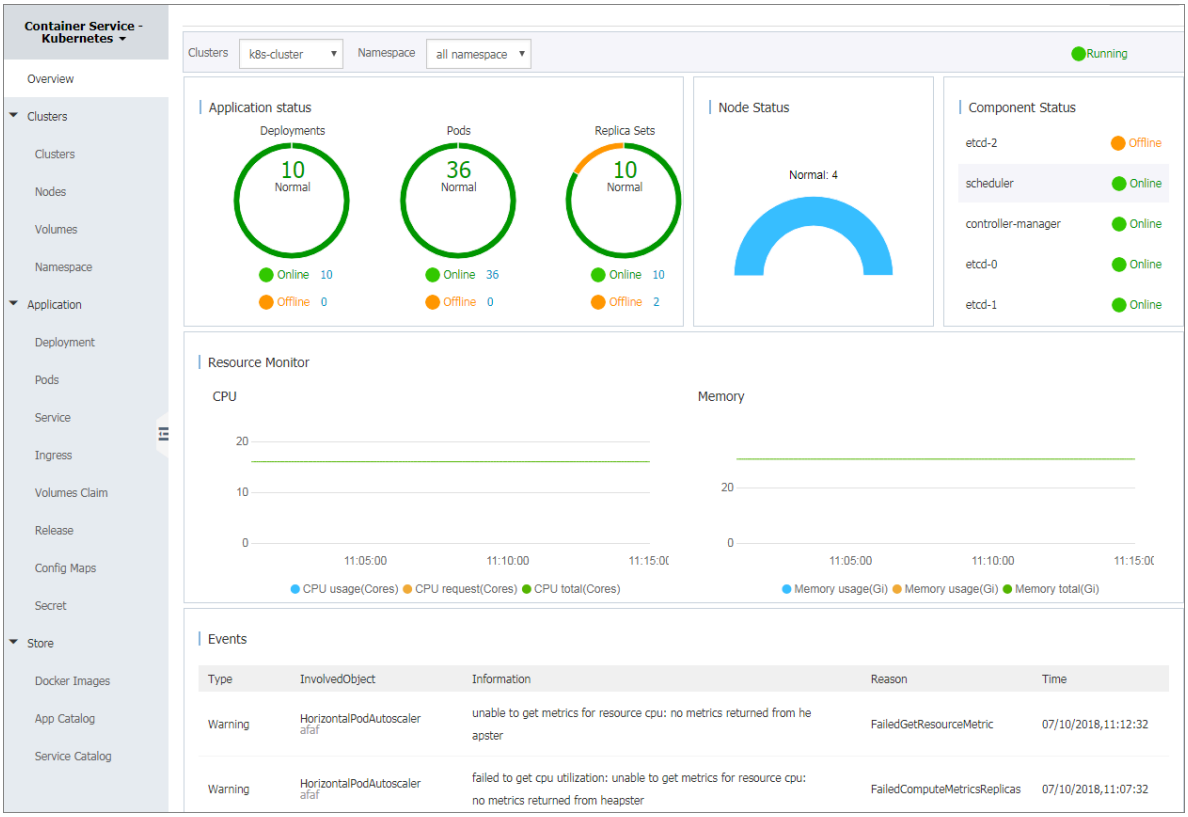
You can view the application status, component status, and resource monitoring charts on the Overview page of Alibaba Cloud Container Service Kubernetes clusters, which allows you to quickly understand the health status of clusters.

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.



1.3 Application Management

1.3.1 Simplify Kubernetes application deployment by using Helm

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

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

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

Basic concepts of Helm

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

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

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

Helm components

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

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

Use Helm to deploy applications

Prerequisites

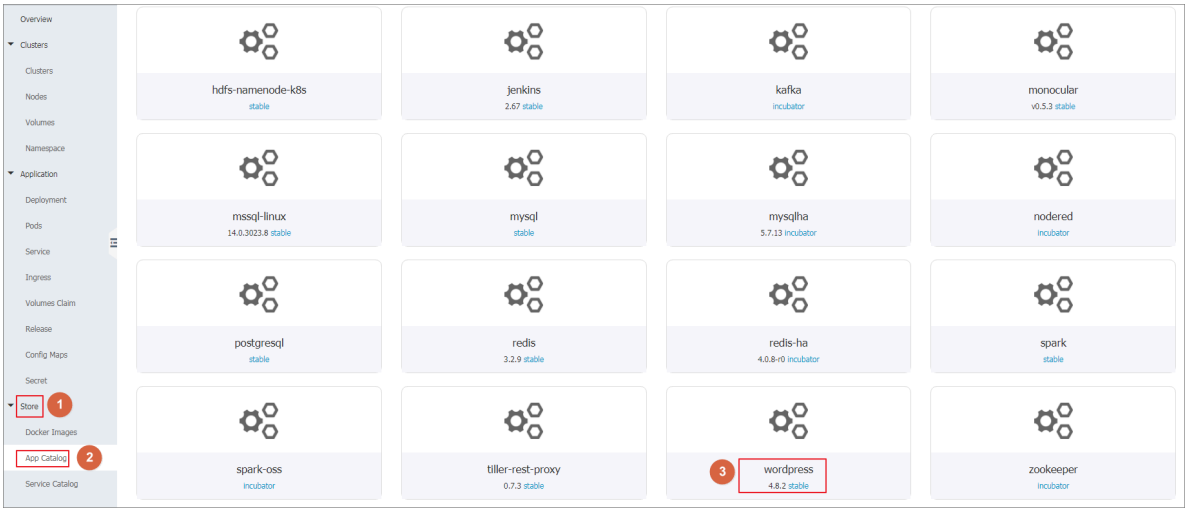
- Before using Helm to deploy an application, create a Kubernetes cluster in Alibaba Cloud Container Service. For more information, see *Create a Kubernetes cluster*.

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

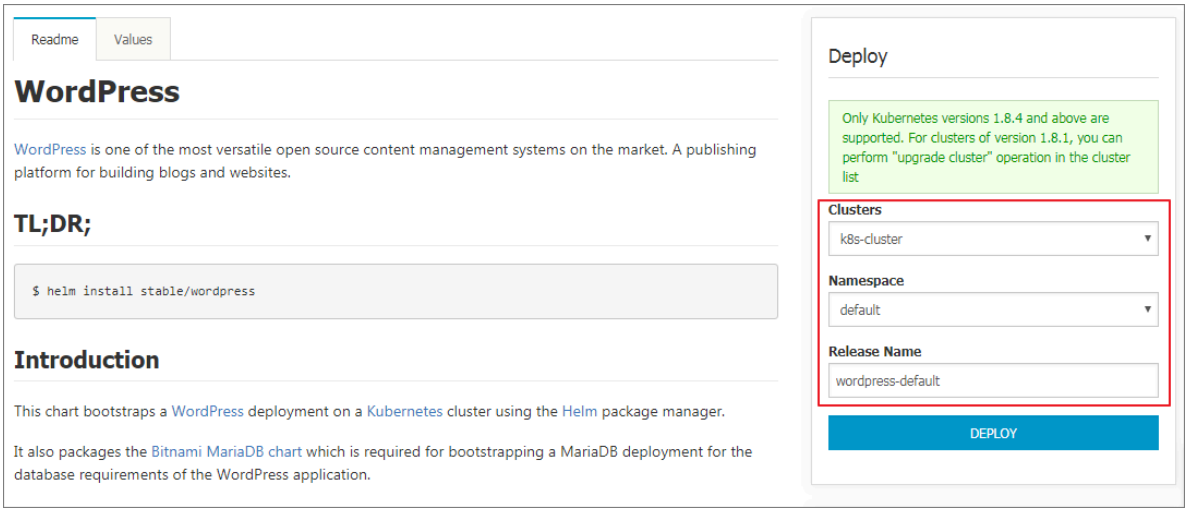
  Only clusters whose Kubernetes version is 1.8.4 or later are supported. For clusters whose Kubernetes version is 1.8.1, upgrade the cluster on the Cluster List page.

Deploy applications in Container Service console

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



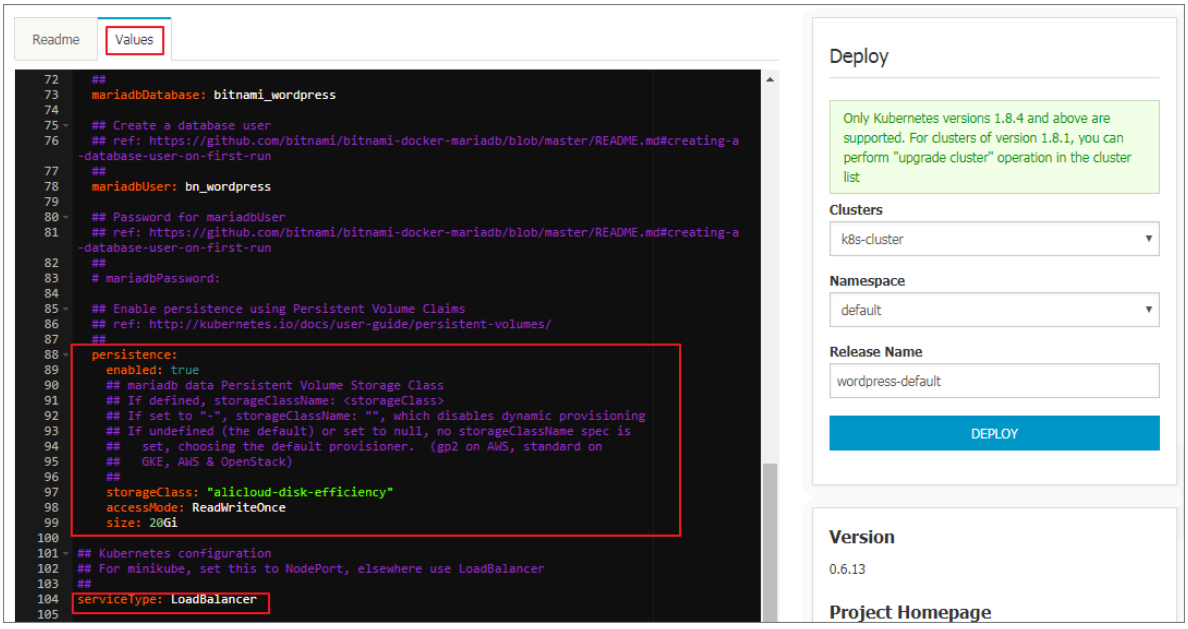
4. Enter the basic information for the deployment on the right.
   - Clusters: Select the cluster in which the application is to be deployed.
   - Namespace: Select the namespace. default is selected by default.
   - Release Name: Enter the release name for the application. Enter test in this example.



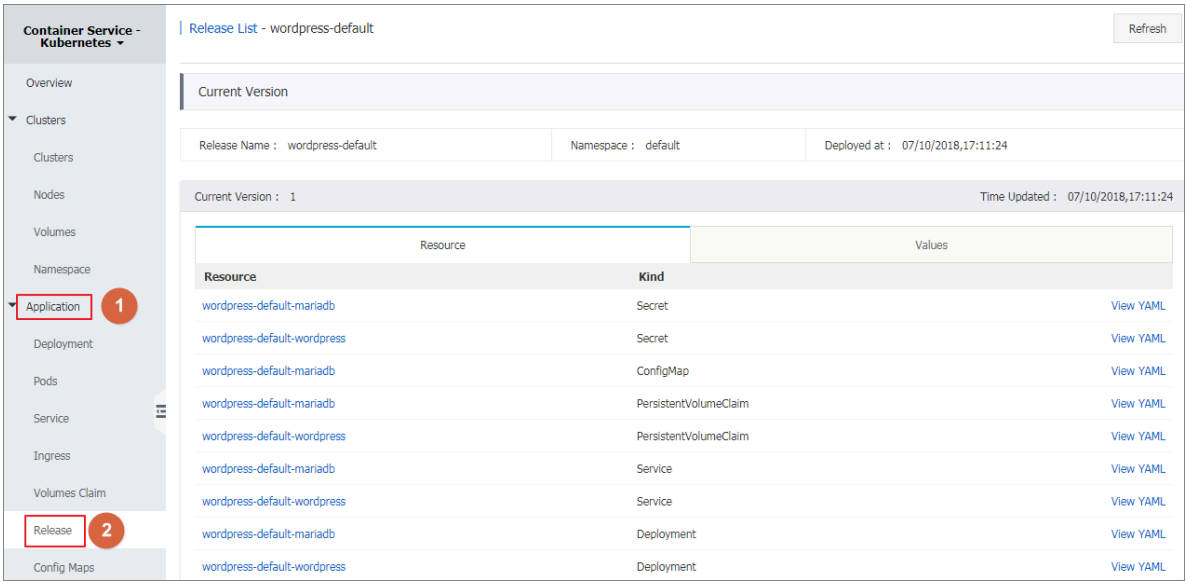
5. Click the Values tab to modify the configurations.

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

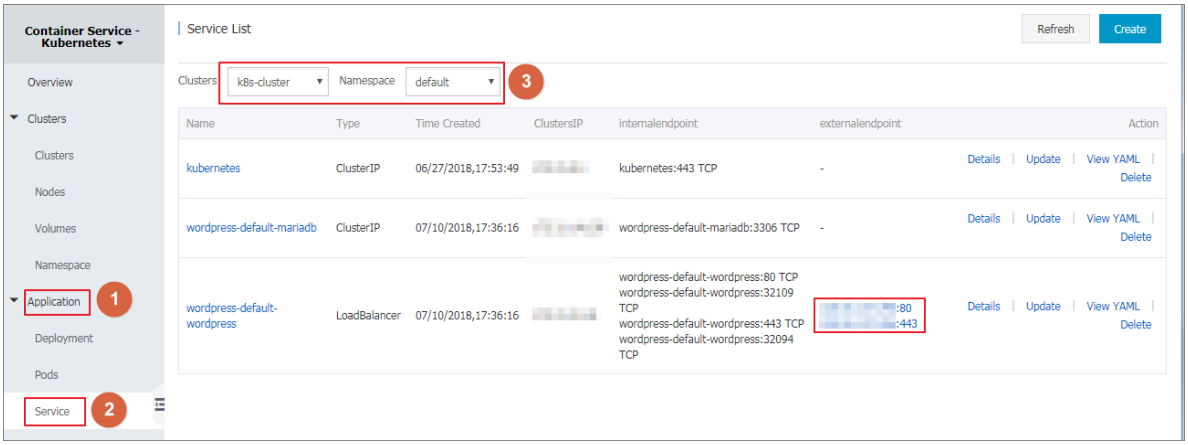
> **Note:**
> You need to create a persistent storage volume (PV) of cloud disk in advance. The capacity of the PV cannot be less than the value defined by the PVC.



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 . loadBalanc er . ingress [ 0 ]. ip }')
```

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

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

```
echo Username : user
```

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

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

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

Deploy Spark by using Helm

Use Helm to deploy Spark for processing big data.

Enter the following command:

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

The result is as follows:

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

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

```
helm   list
kubectl   get   svc
```

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

```
kubectl   get   pod
```

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

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

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

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

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

The result is as follows:

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

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

```
kubectl get pod
```

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

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

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

Use third-party chart repository

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

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

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

References

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

1.4 Namespaces

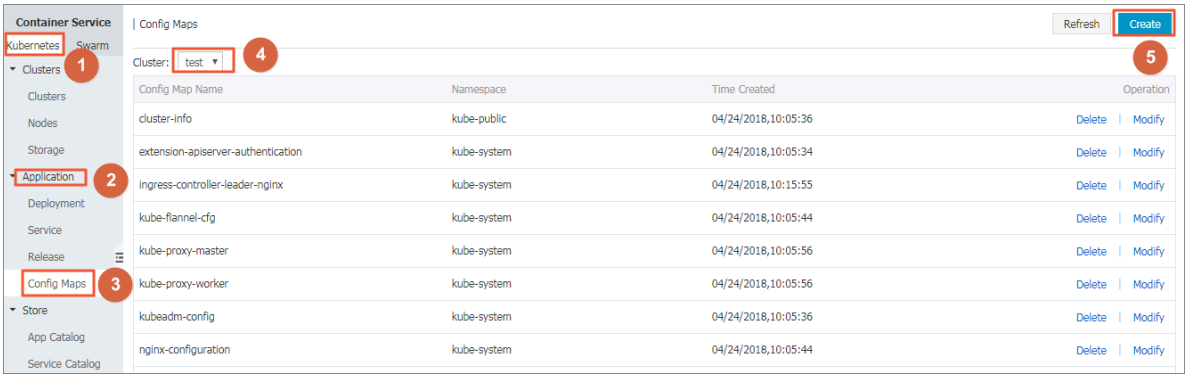
1.5 Config map

1.5.1 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

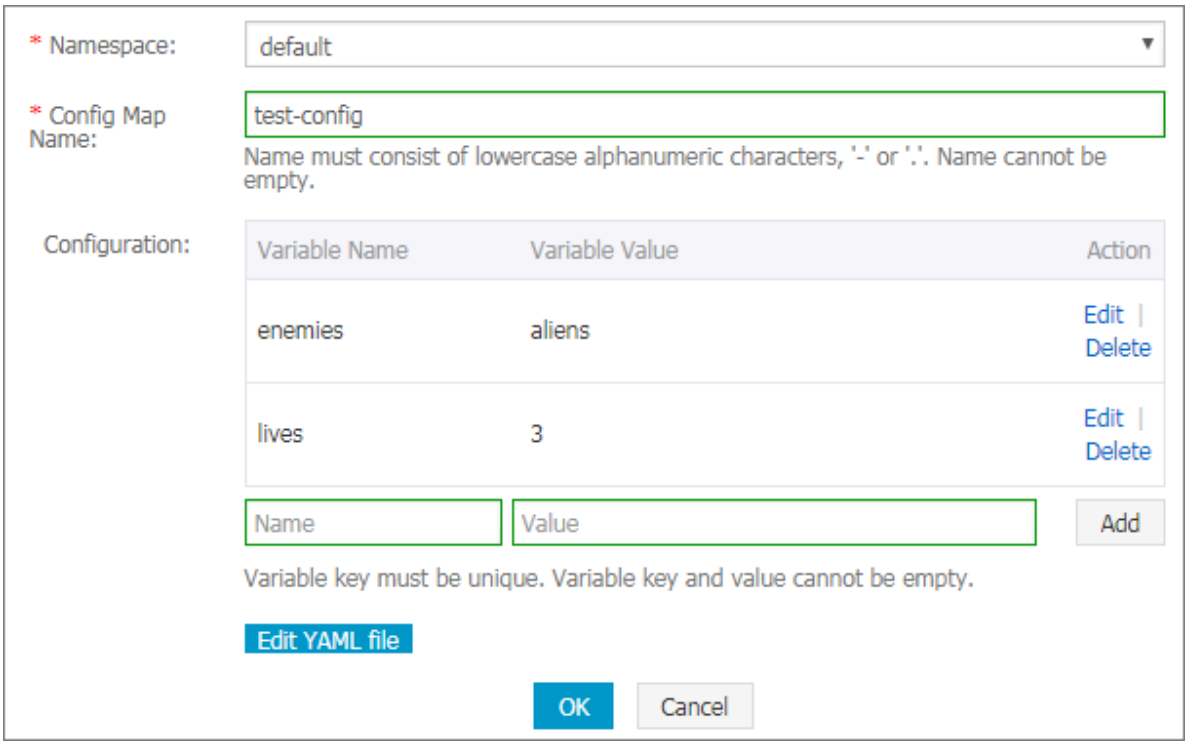
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Application > Config Maps in the left-side navigation pane.
3. Select the cluster from the Cluster drop-down list. Click Create in the upper-right corner.



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.

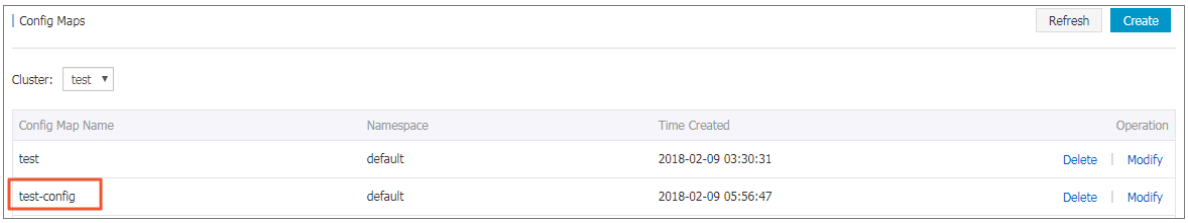
Other resource objects must reference the config map name to obtain the configuration information.

- 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.

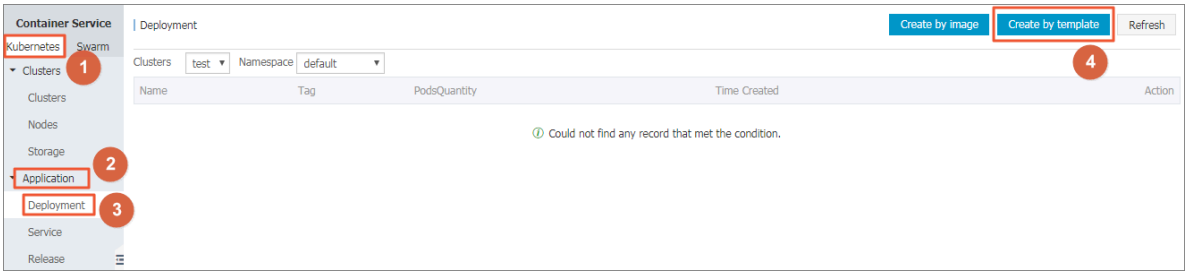
5. You can view the config map test-config on the Config Maps page after clicking OK.



Create a config map by using a template

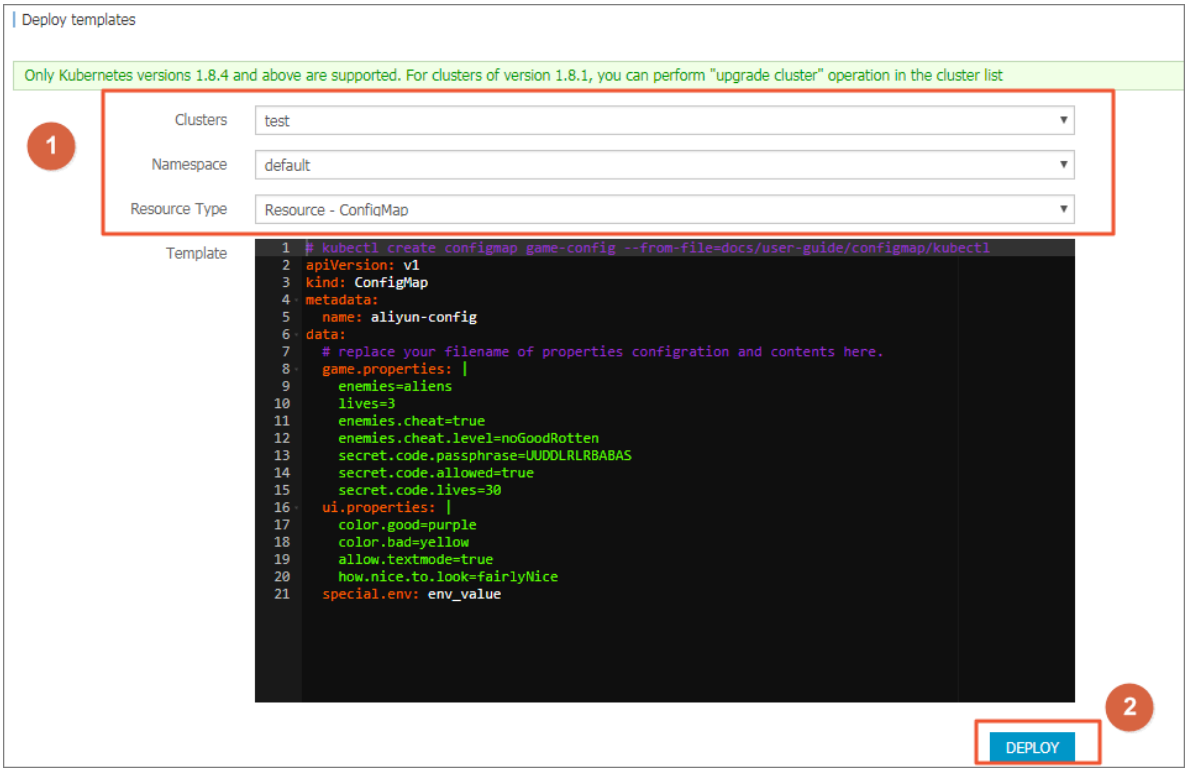
1. Log on to the *Container Service console*.
2. Under Kubernetes, click Application > > Deploymentin the left-side navigation pane.

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

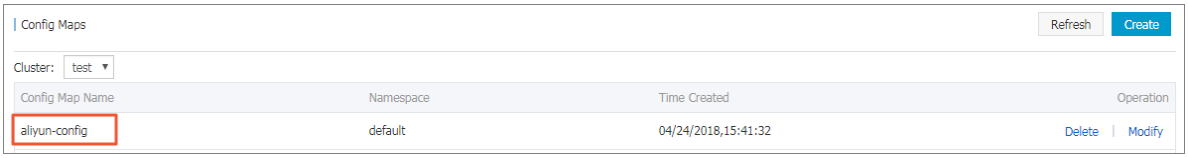


4. On the Deploy templates page, complete the settings and then click DEPLOY.

   - 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.



1.5.2 Use a config map in a pod

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

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

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

Limits

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

Create a config map

In this example, create a config map special-config, which includes two key-value pairs: `SPECIAL_LE VEL : very` and `SPECIAL_TY PE : charm`.

Create a config map by using an orchestration template

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

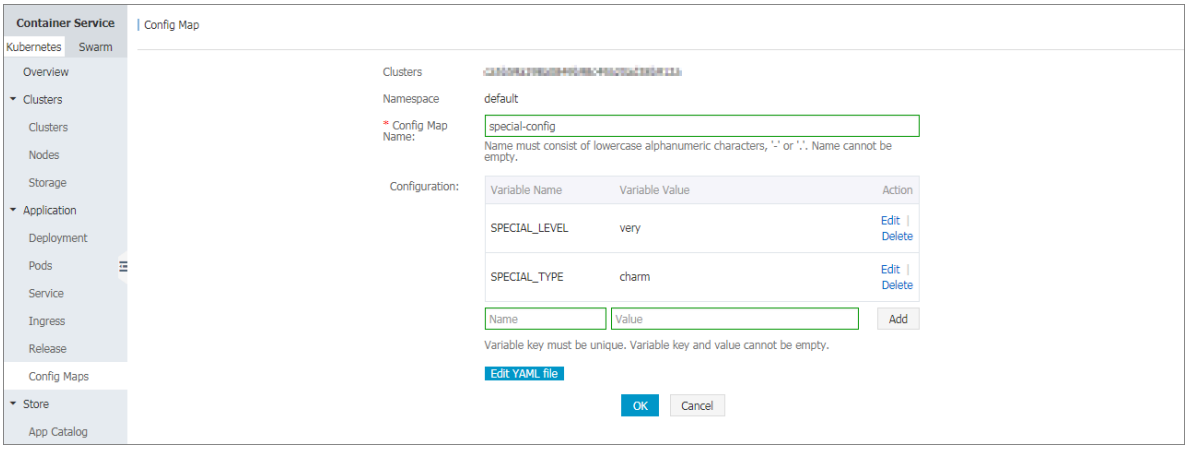
   You can use the following YAML sample template to create a config map.

```
apiVersion : v1
 kind : ConfigMap
 metadata :
   name : special - config
   namespace : default
 data :
   SPECIAL_LE VEL : very
```

```
    SPECIAL_TY  PE :  charm
```

Create a config map on Config Maps page

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

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

   See the following orchestration example:

```
apiVersion :  v1
 kind :  Pod
 metadata :
   name :  config - pod - 1
 spec :
   containers :
```

```
    - name : test - container
      image : busybox
      command : [ "/ bin / sh ", "- c ", " env " ]
      env :
       - name : SPECIAL_LE VEL_KEY
         valueFrom : ## Use valueFrom to specify env to
  reference the value of the config map .
           configMapK eyRef :
             name : special - config ## The referenced
config map name .
             key : SPECIAL_LE VEL ## The referenced config
  map key .
   restartPol icy : Never
```

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

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

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

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

   See the following orchestration example:

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

Use a config map to configure command line parameters

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

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

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

   See the following orchestration example:

```
apiVersion :  v1
 kind :  Pod
 metadata :
   name :  config - pod - 3
 spec :
   containers :
    -  name :  test - container
       image :  busybox
       command : [ "/ bin / sh ", "- c ", " echo  $( SPECIAL_LE
VEL_KEY ) $( SPECIAL_TY  PE_KEY )" ]
       env :
        -  name :  SPECIAL_LE  VEL_KEY
           valueFrom :
             configMapK  eyRef :
               name :  special - config
               key :  SPECIAL_LE  VEL
        -  name :  SPECIAL_TY  PE_KEY
           valueFrom :
             configMapK  eyRef :
               name :  special - config
               key :  SPECIAL_TY  PE
   restartPol  icy :  Never
```

   The output after running the pod is as follows:

```
very   charm
```

Use a config map in data volumes

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

   You can also use a config map in data volumes. Specifying the config map name under volumes stores the key-value pair data to the mountPath directory (*/ etc /*

config in this example). 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 .
      volumeMoun ts :
     - name : config - volume
       mountPath : / etc / config
   volumes :
    - name : config - volume
      configMap :
        name : special - config
   restartPol icy : Never
```

Keys of the config map are output after running the pod.

```
SPECIAL_TY PE
SPECIAL_LE VEL
```

1.5.3 Update a config map

You can modify the configurations of a config map.

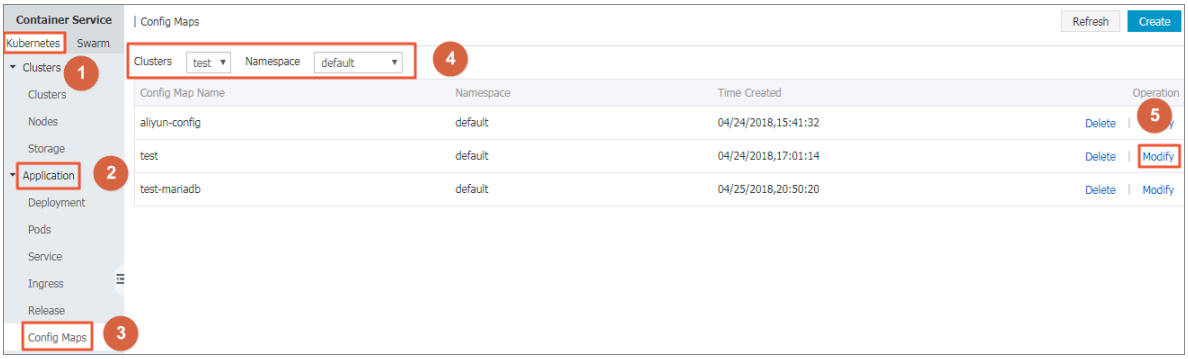
Note:

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

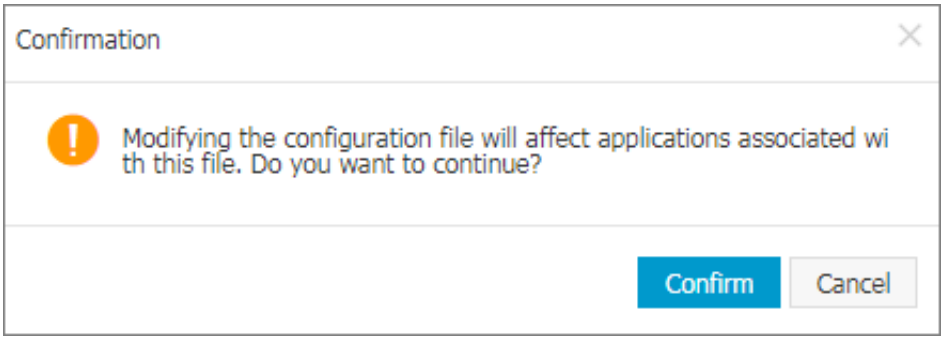
Update a config map on Config Maps page

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

3. Select the cluster and namespace from the Clusters and Namespace drop-down lists. Click Modify at the right of the config map.

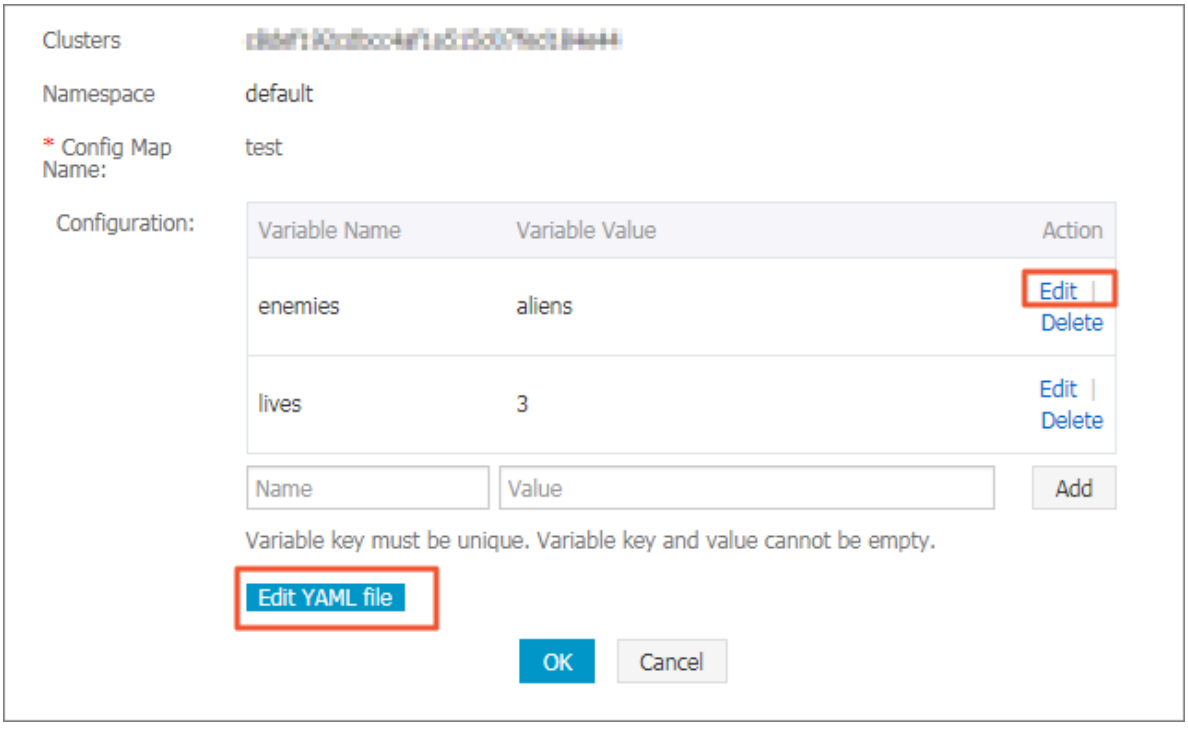


4. Click Confirm in the displayed dialog box.



5. Modify the configurations.

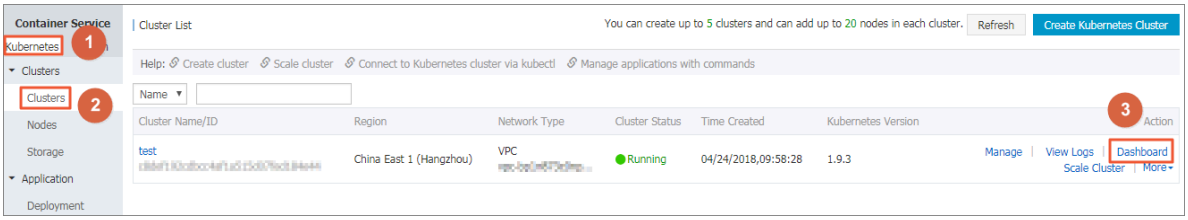
   - Click Edit on the right of the configuration you want to modify. Update the configuration and then click Save.
   - You can also click Edit YAML file. Click OK after making the modifications.



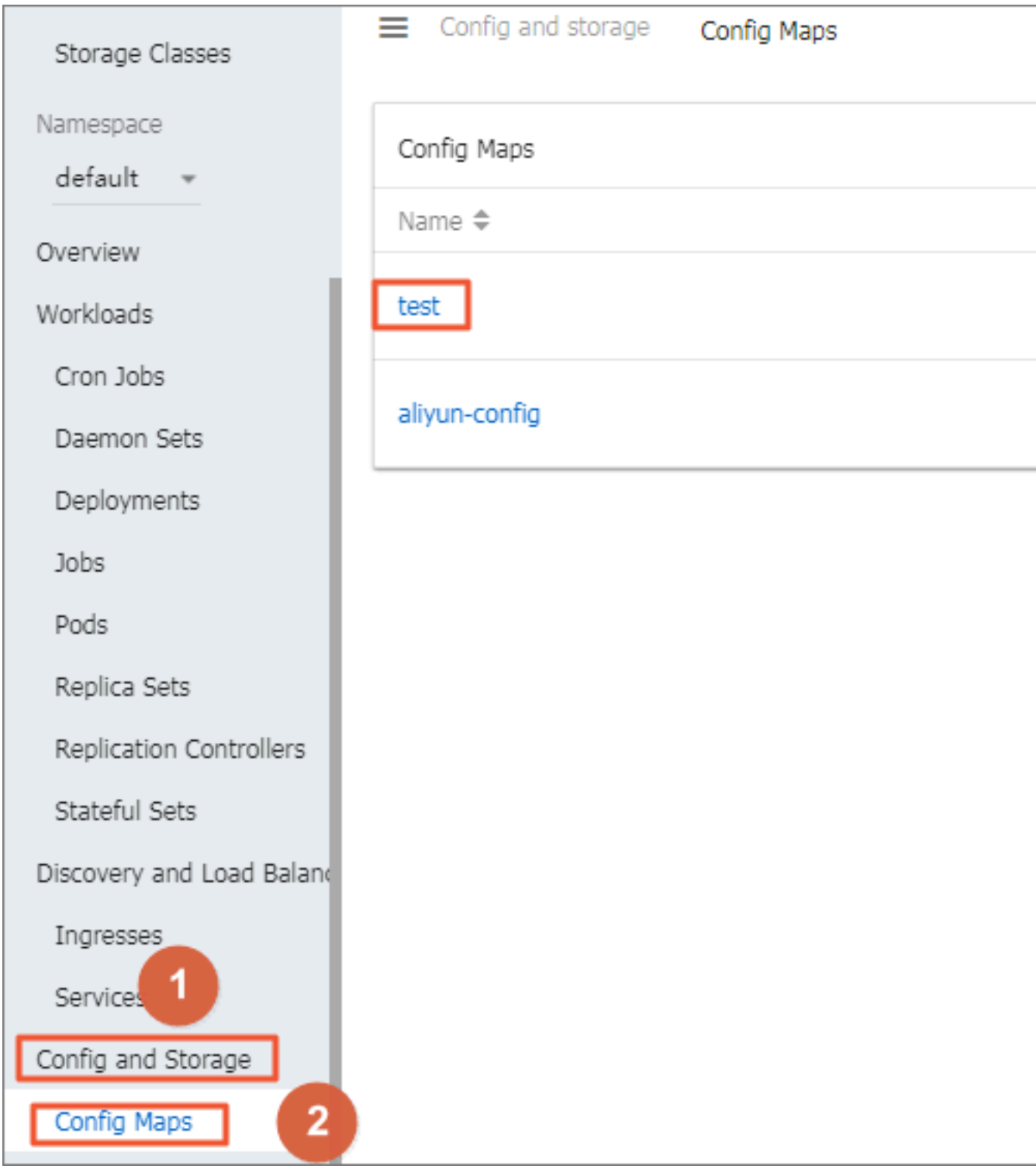
6. After modifying the configurations, click OK.

Update a config map in Kubernetes dashboard

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



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



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

1.6 Secrets

1.7 App catalog

1.8 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:

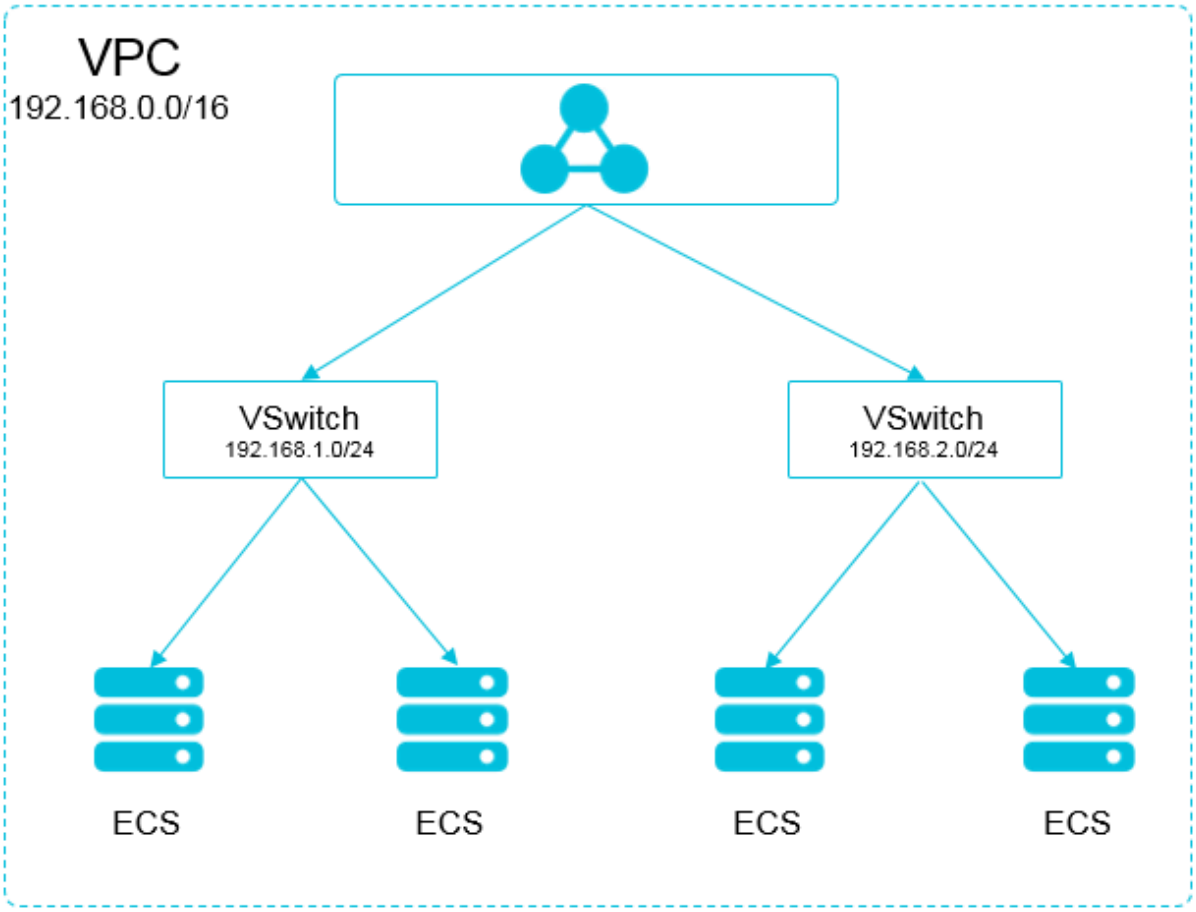
VPC CIDR block

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.



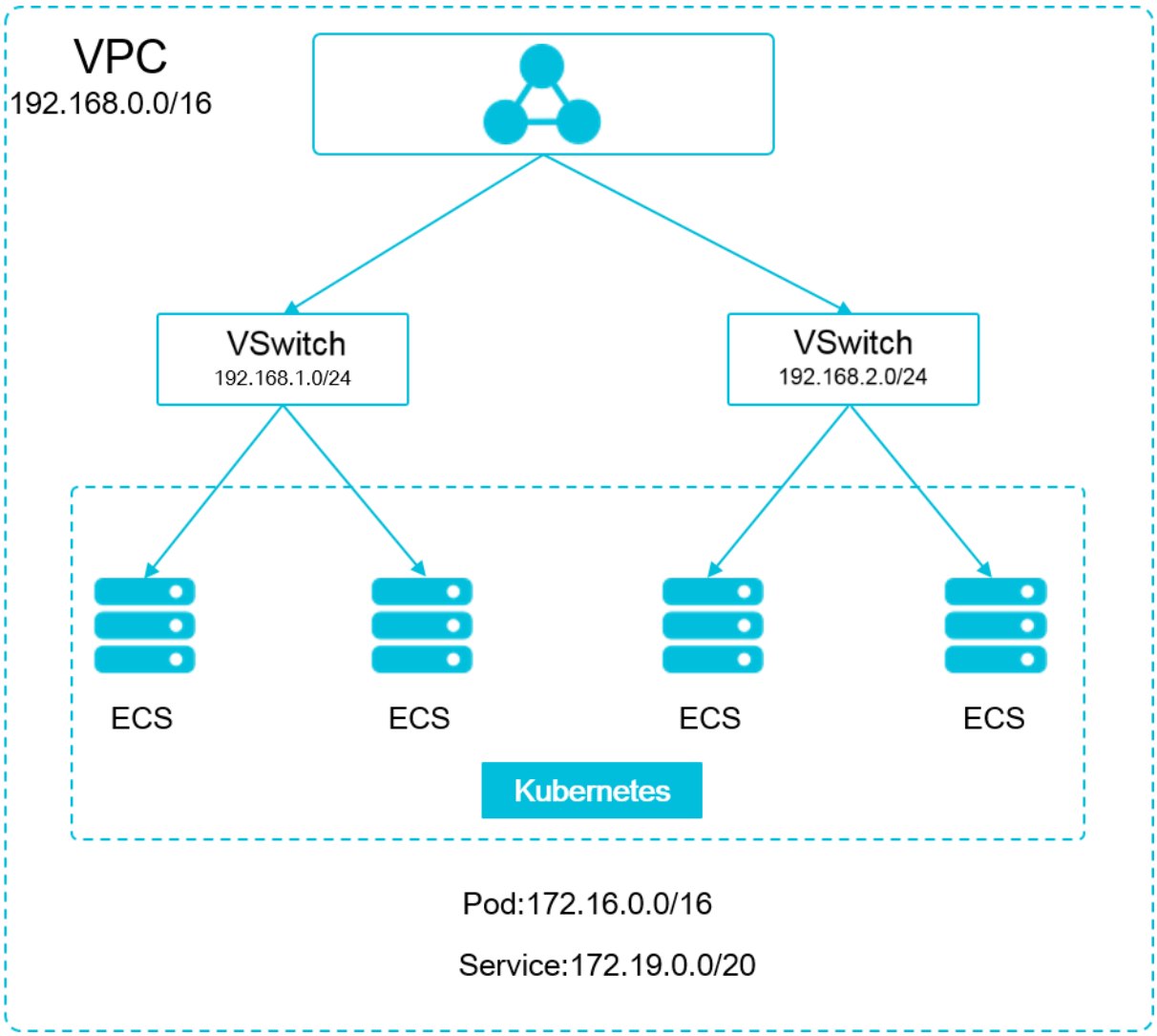
Pod CIDR block

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

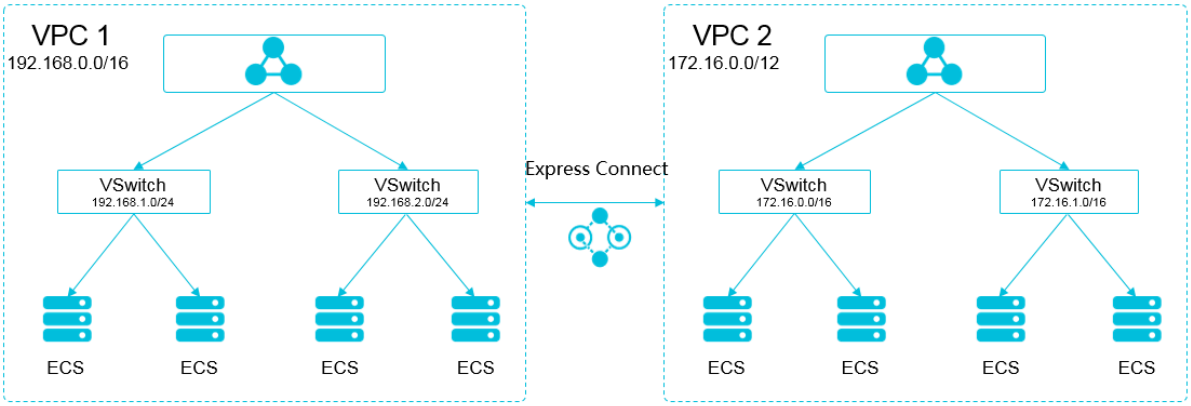
Create multiple Kubernetes clusters under one VPC. In the default network mode (Flannel), the pod message needs to be routed by using VPC, and Container Service automatically configures the route table to each pod CIDR block on the VPC route. The pod CIDR blocks of all the Kubernetes clusters cannot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

1.9 Server Load Balancer

1.9.1 Access services by using Server Load Balancer

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

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

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

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

```
    image :  registry - vpc . cn - hangzhou . aliyuncs . com / acs /
cloud - controller - manager - amd64 : v1 . 9 . 3
```

Operate by using command line

1. Create an Nginx application by using command line.

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

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

```
root @ master  #  kubectl   expose   deployment   nginx  -- port =
80  -- target - port = 80  -- type = LoadBalanc  er
root @ master  #  kubectl   get   svc
NAME                   CLUSTER - IP        EXTERNAL - IP
PORT ( S )                    AGE
nginx                  172 . 19 . 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.

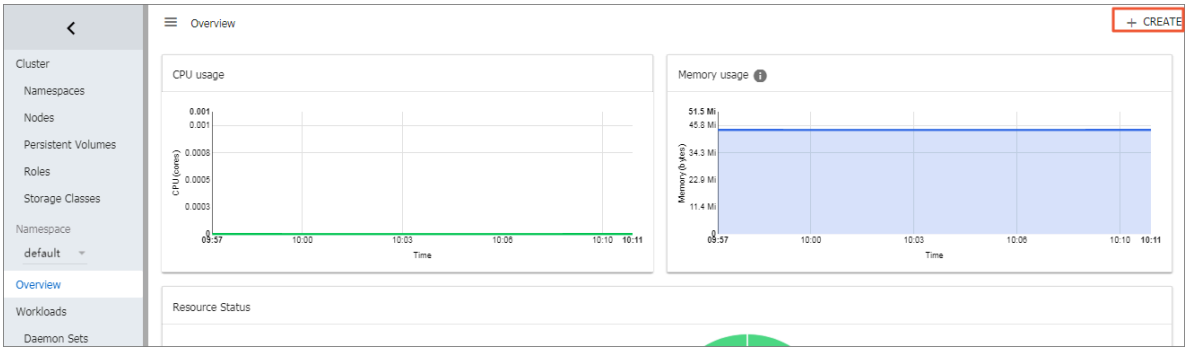
Operate by using Kubernetes dashboard

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 :  LoadBalanc  er
```

2. Log on to the *Container Service console*. Click Dashboard at the right of a cluster.

3. Click CREATE in the upper-right corner to create an application.

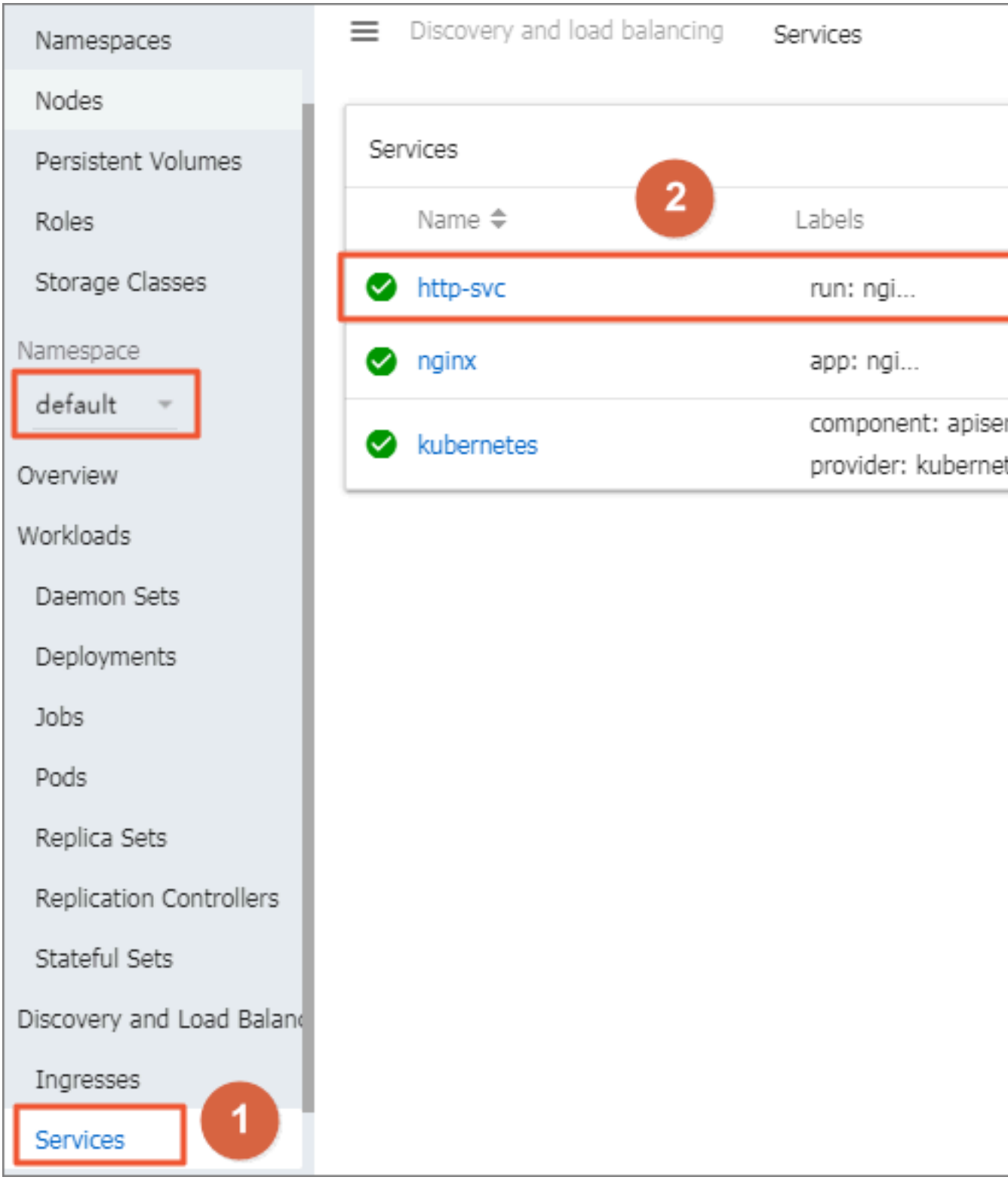


4. Click the CREATE FROM FILE tab. and then upload the `nginx-svc.yml` file you saved.
5. Click UPLOAD.

   A Nginx application specified by Alibaba Cloud Server Load Balancer instance is created. The service name is `http-svc`.

6. Select default under Namespace in the left-side navigation pane. Click `Services` in the left-side navigation pane.

   You can view the created Nginx service `http - svc` and the Server Load Balancer address `http :// 114 . 55 . 79 . 24 : 80` .



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

More information

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

Annotations

Alibaba Cloud supports a lot 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 :
  annotation s :
    service . beta . kubernetes . io / alicloud - loadbalanc er -
address - type : intranet
    service . beta . kubernetes . io / alicloud - loadbalanc er - id
: your - loadbalanc er - id
  labels :
    run : nginx
  name : nginx
  namespace : default
spec :
  ports :
 - name : web
    port : 80
    protocol : TCP
    targetPort : 80
  selector :
    run : nginx
  sessionAff inity : None
  type : LoadBalanc er
```

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 :
  annotation s :
    service . beta . kubernetes . io / alicloud - loadbalanc er -
cert - id : your - cert - id
    service . beta . kubernetes . io / alicloud - loadbalanc er -
protocol - port : " https : 443 "
  labels :
```

```
    run :  nginx
  name :  nginx
  namespace :  default
spec :
  ports :
 -  name :  web
    port :  443
    protocol :  TCP
    targetPort :  443
  selector :
    run :  nginx
  sessionAff  inity :  None
  type :  LoadBalanc  er
```

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.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-address-type	The value is Internet or intranet.	internet
service.beta.kubernetes.io/alicloud-loadbalancer-slb-network-type	Server Load Balancer network type. The value is classic or VPC.	classic
service.beta.kubernetes.io/alicloud-loadbalancer-charge-type	The value is paybytraffic or paybybandwidth.	paybybandwidth
service.beta.kubernetes.io/alicloud-loadbalancer-id	The Server Load Balancer instance ID. Specify an existing Server Load Balance with the loadbalancer-id, and the existing listener is overwritten. Server Load Balancer is not deleted when the service is deleted.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-backend-label	Use label to specify which nodes are mounted to the Server Load Balancer backend.	N/A

Annotation	Description	Default value
service.beta.kubernetes.io/alicloud-loadbalancer-region	The region in which Server Load Balancer resides.	N/A
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 *../../SP_23/DNSLB11870158/EN-US_TP_4205.dita#doc_api_Slb_CreateLoadBalancerTCPListener*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-uri	See *../../SP_23/DNSLB11870158/EN-US_TP_4205.dita#doc_api_Slb_CreateLoadBalancerTCPListener*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-port	See *../../SP_23/DNSLB11870158/EN-US_TP_4205.dita#doc_api_Slb_CreateLoadBalancerTCPListener*.	
service.beta.kubernetes.io/alicloud-loadbalancer-healthy-threshold	See *../../SP_23/DNSLB11870158/EN-US_TP_4205.dita#doc_api_Slb_CreateLoadBalancerTCPListener*.	
service.beta.kubernetes.io/alicloud-loadbalancer-unhealthy-threshold	See *../../SP_23/DNSLB11870158/EN-US_TP_4205.dita#doc_api_Slb_CreateLoadBalancerTCPListener*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval	See *../../SP_23/DNSLB11870158/EN-US_TP_4205.dita#doc_api_Slb_CreateLoadBalancerTCPListener*.	
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout	See *../../SP_23/DNSLB11870158/EN-US_TP_4205.dita#doc_api_Slb_CreateLoadBalancerTCPListener*.	

Annotation	Description	Default value
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout	See ../../SP_23/DNSLB11870158/EN-US_TP_4205.dita#doc_api_Slb_CreateLoadBalancerTCPListener.	

1.9.2 Support for Ingress

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

Prerequisites

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

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

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

Simple routing service

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

```
root @ master  #  cat  << EOF  |  kubectl   create  - f  -
apiVersion :  extensions / v1beta1
kind :  Ingress
metadata :
  name :  simple
  annotation  s :
    nginx . ingress . kubernetes . io / rewrite - target : /
spec :
  rules :
 -  http :
      paths :
```

```
      -  path : / svc
         backend :
           serviceNam  e :  http - svc
           servicePor  t :  80
EOF
root @ master  #  kubectl   get   ing
NAME              HOSTS           ADDRESS            PORTS       AGE
simple           *              101 . 37 . 192 . 211     80
11s
```

Now visit `http :// 101 . 37 . 192 . 211 / svc` to access the Nginx service.

Simple fanout routing based on domain names

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

```
root @ master  #  cat  << EOF  |  kubectl   create  - f  -
apiVersion :  extensions / v1beta1
kind :  Ingress
metadata :
  name :  simple - fanout
spec :
  rules :
 -  host :  foo . bar . com
    http :
      paths :
     -  path : / foo
        backend :
          serviceNam  e :  http - svc1
          servicePor  t :  80
     -  path : / bar
        backend :
          serviceNam  e :  http - svc2
          servicePor  t :  80
 -  host :  foo . example . com
    http :
      paths :
     -  path : / film
        backend :
          serviceNam  e :  http - svc3
          servicePor  t :  80
EOF
root @ master  #  kubectl   get   ing
NAME              HOSTS           ADDRESS            PORTS       AGE
simple - fanout    *              101 . 37 . 192 . 211     80
  11s
```

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

Note:

- In a production environment, point the domain name to the preceding returned address `101 . 37 . 192 . 211` .
- In a testing environment, you can modify the `hosts` file to add a domain name mapping rule.

```
101 . 37 . 192 . 211   foo . bar . com
101 . 37 . 192 . 211   foo . example . com
```

Default domain name of simple routing

It does not matter if you do not have the domain name address. Container Service binds a default domain name for Ingress service. You can use this default domain name to access the services. The domain name is in the format of `*.[ cluster - id ].[ region - id ]. alicontain er . com` . You can obtain the address on the cluster Basic Information page in the console.

Use the following configuration to expose two services with the default domain name.

```
root @ master # cat << EOF | kubectl create - f -
apiVersion : extensions / v1beta1
kind : Ingress
metadata :
  name : shared - dns
spec :
  rules :
 - host : foo .[ cluster - id ].[ region - id ]. alicontain er .
com ## Replace with the default service access domain
name of your cluster .
    http :
      paths :
     - path : /
        backend :
          serviceNam e : http - svc1
          servicePor t : 80
 - host : bar .[ cluster - id ].[ region - id ]. alicontain er .
com ## Replace with the default service access domain
name of your cluster .
    http :
      paths :
     - path : /
        backend :
          serviceNam e : http - svc2
          servicePor t : 80
EOF
root @ master # kubectl get ing
NAME              HOSTS           ADDRESS            PORTS       AGE
shared - dns     foo .[ cluster - id ].[ region - id ]. alicontain
er . com , bar .[ cluster - id ].[ region - id ]. alicontain er .
com               47 . 95 . 160 . 171     80          40m
```

Then, you can access the `http - svc1` service by using `http :// foo .[ cluster - id ].[ region - id ]. alicontain er . com /`and access the

`http-svc2` service by using `http://bar.[cluster-id].[region-id].alicontainer.com`.

Configure a safe routing service

Management of multiple certificates is supported to provide security protection for your services.

1. Prepare your service certificate.

   If no certificate is available, generate a test certificate in the following method:

   > **Note:**
   > The domain name must be consistent with your Ingress configuration.

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

   The above command generates a certificate file `tls.crt` and a private key file `tls.key`.

   Create a Kubernetes secret named `foo.bar` using the certificate and private key. The secret must be referenced when you create the Ingress.

   ```
   root@master# kubectl create secret tls foo.bar --key tls.key --cert tls.crt
   ```

2. Create a safe Ingress service.

   ```
   root@master# cat << EOF | kubectl create -f -
   apiVersion: extensions/v1beta1
   kind: Ingress
   metadata:
     name: tls-fanout
   spec:
     tls:
     - hosts:
       - foo.bar.com
       secretName: foo.bar
     rules:
     - host: foo.bar.com
       http:
         paths:
         - path: /foo
           backend:
             serviceName: http-svc1
             servicePort: 80
         - path: /bar
           backend:
             serviceName: http-svc2
             servicePort: 80
   EOF
   ```

```
root @ master # kubectl get ing
NAME HOSTS ADDRESS PORTS
AGE
tls - fanout * 101 . 37 . 192 . 211 80
   11s
```

3. Follow the notes in Simple fanout routing based on domain names to configure the `hosts` file or set the domain name to access the TLS service.

   You can access the `http - svc1` service by using `http :// foo . bar . com / foo` and access the `http - svc2` service by using `http :// foo . bar . com / bar` .

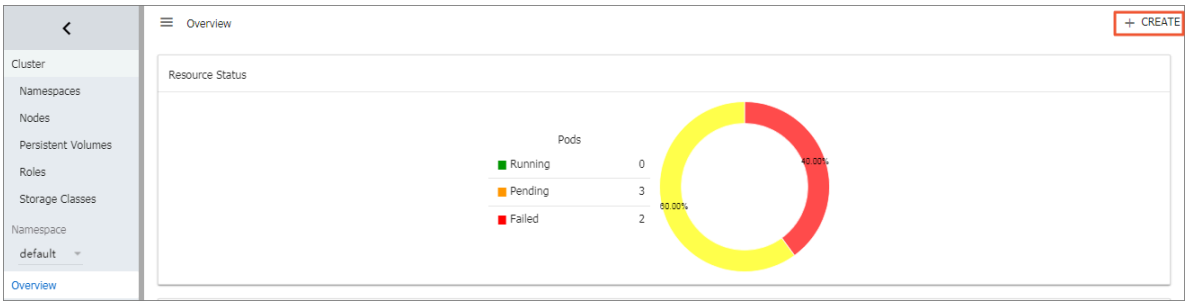
   You can also access the HTTPS service by using HTTP. By default, Ingress redirects HTTP access configured with HTTPS to the HTTPS address. Therefore, access to `http :// foo . bar . com / foo` will be automatically redirected to `https :// foo . bar . com / foo` .

Deploy Ingress in Kubernetes dashboard

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

```
apiVersion : extensions / v1beta1
kind : Ingress
metadata :
  name : simple
spec :
  rules :
 - http :
     paths :
     - path : / svc
       backend :
         serviceNam e : http - svc
         servicePor t : 80
```

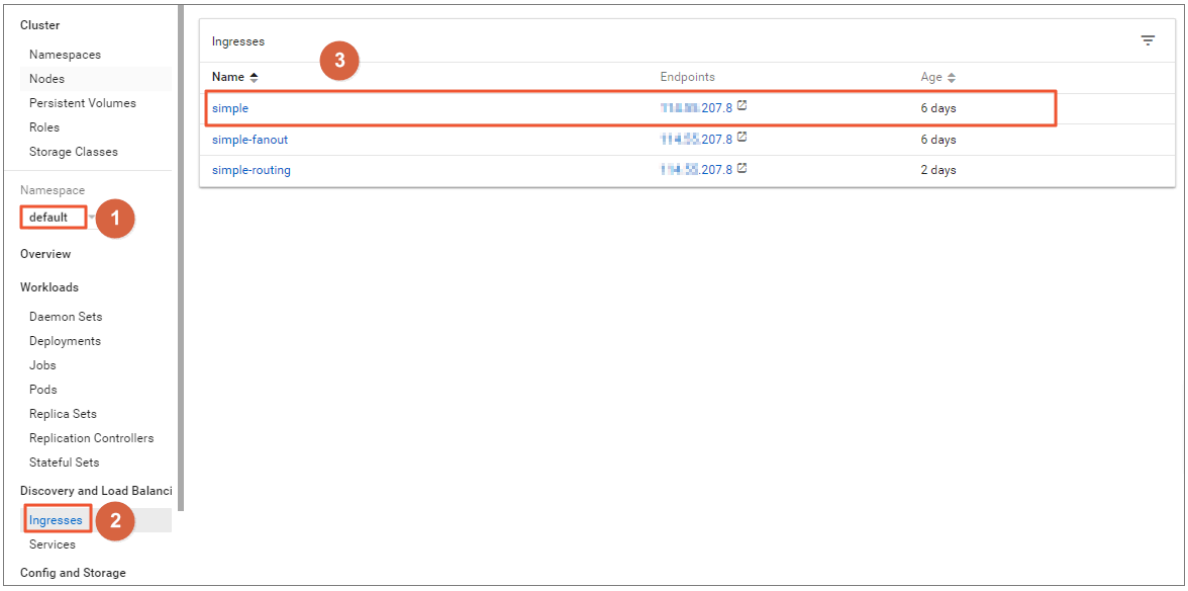
2. Log on to the #########. Under Kubernetes, click Clusters in the left-side navigation pane. Click Dashboardat the right of the cluster to enter the Kubernetes dashboard.
3. Click CREATE in the upper-right corner to create an application.



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

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

   You can view the created Ingress resource and its access address `http :// 118 . 178 . 174 . 161 / svc` .



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

1.10 Storage

1.10.1 Use Alibaba Cloud cloud disks

You can use the Alibaba Cloud cloud disk storage volumes in Alibaba Cloud Container Service Kubernetes clusters.

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

- *Static storage volumes*

  You can use the cloud disk static storage volumes by:

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

- *Dynamic storage volumes*

Note:

The following requirements are imposed on the created cloud disk capacity:

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

Static storage volumes

You can use Alibaba Cloud cloud disk storage volumes by using the volume method or PV/PVC.

Prerequisites

Before using cloud disk data volumes, you must create cloud disks in the Elastic Compute Service (ECS) console. For how to create cloud disks, see *Create a cloud disk*.

Instructions

- **The cloud disk is not a shared storage and can only be mounted by one pod at the same time.**
- **Apply for a cloud disk and obtain the disk ID before using cloud disk storage volumes. See *Create a cloud disk*.**
- **volumeId: The disk ID of the mounted cloud disk, which must be the same as volumeName and PV Name.**
- **Only the cluster node that is in the same zone as the cloud disk can mount the cloud disk.**

Use volume method

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

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-disk-deploy
spec:
```

```
  replicas :  1
  template :
    metadata :
      labels :
        app :  nginx
    spec :
      containers :
       -  name :  nginx - flexvolume - disk
          image :  nginx
          volumeMoun  ts :
           -  name : " d - bp1j17ifxf  asvts3tf40 "
              mountPath : "/ data "
      volumes :
       -  name : " d - bp1j17ifxf  asvts3tf40 "
          flexVolume :
            driver : " alicloud / disk "
            fsType : " ext4 "
            options :
              volumeId : " d - bp1j17ifxf  asvts3tf40 "
```

Use PV/PVC

Step 1. Create a cloud disk type PV

You can create the cloud disk type PV in the Container Service console or by using the yaml file.

Create PV by using yaml file

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

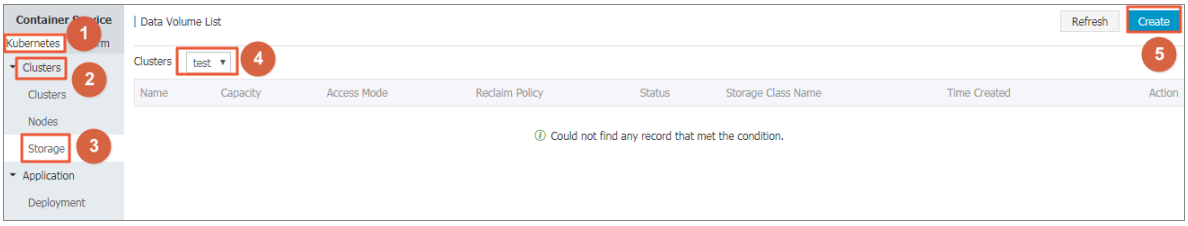
Note:

The PV name must be the same as the Alibaba Cloud cloud disk ID.

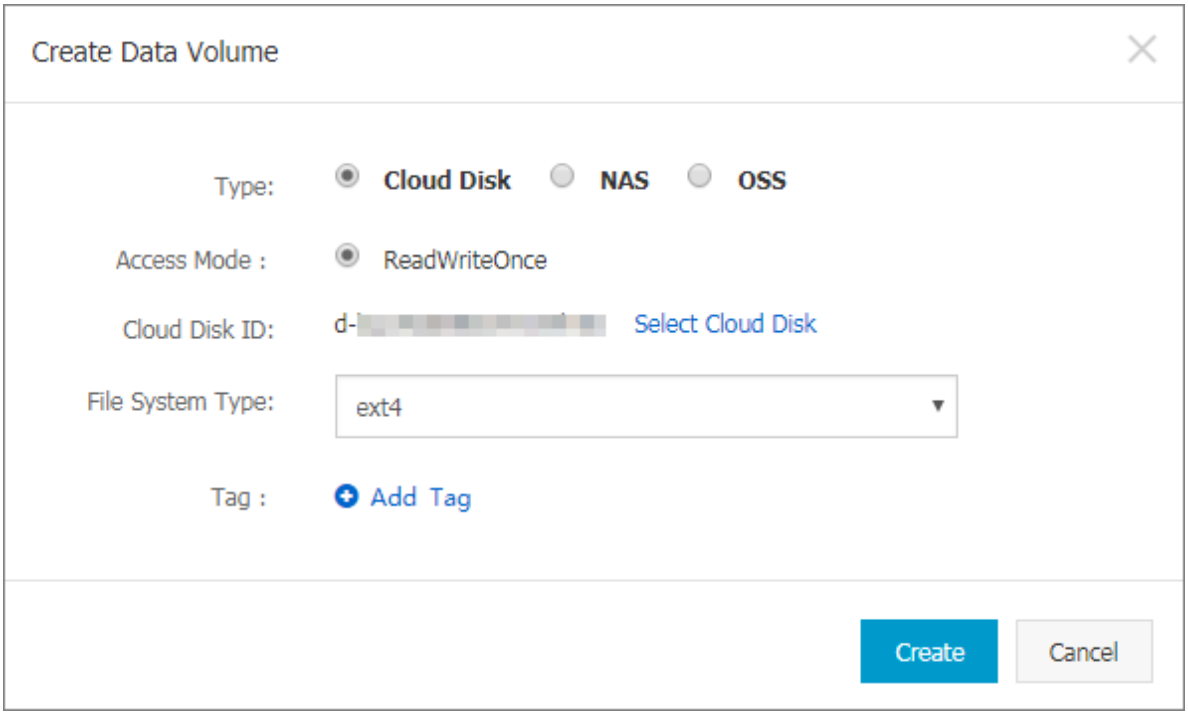
```
apiVersion :  v1
kind :  Persistent  Volume
metadata :
  name :  d - bp1j17ifxf  asvts3tf40
  labels :
    failure - domain . beta . kubernetes . io / zone :  cn - hangzhou
- b
    failure - domain . beta . kubernetes . io / region :  cn -
hangzhou
spec :
  capacity :
    storage :  20Gi
  storageCla  ssName :  disk
  accessMode  s :
   -  ReadWriteO  nce
  flexVolume :
    driver : " alicloud / disk "
    fsType : " ext4 "
    options :
      volumeId : " d - bp1j17ifxf  asvts3tf40 "
```

Create cloud disk data volumes in Container Service console

1. Log on to the *Container Service console*.
2. Under Kubernetes, clickClusters > Storagein 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: `cloud disk` in the example.
   - Name: The name of the created data volume. The data volume name must be the same as the Cloud Disk ID.
   - Access Mode: ReadWriteOnce by default.
   - Cloud Disk ID: Select the cloud disk to be mounted and is in the same region and zone as the cluster.
   - File System Type: You can select the data type in which data is stored to the cloud disk. The supported types include ext4, ext3, xfs, and vfat ext4 is selected by default.
   - Tag: Click Add Tag to add tags for this data volume.



5. After the preceding settings, click Create.

Step 2. Create PVC

Use the `disk-pvc.yaml` file to create the PVC.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-disk
spec:
  accessModes:
   - ReadWriteOnce
  storageClassName: disk
  resources:
    requests:
      storage: 20Gi
```

Step 3. Create a pod

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

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

Dynamic storage volumes

Dynamic storage volumes require you to manually create a Storage Class and specify the target type of cloud disk in the PVC by storage Class Name.

Create a StorageClass

```
kind: StorageClass
apiVersion: storage.k8s.io/v1beta1
metadata:
  name: alicloud-disk-common-hangzhou-b
provisioner: alicloud/disk
parameters:
  type: cloud_ssd
  regionid: cn-hangzhou
  zoneid: cn-hangzhou-b
```

Parameters:

- provisioner: configured as Alibaba Cloud/disk, the identifier is created using the Alibaba Cloud Provsioner plug-in.
- type: identifies the cloud disk type, supports cloud, cloud_efficiency, cloud_ssd, and available types. To improve efficiency, tries to create SSD, and common cloud disk until it is created successfully.
- regionid: the region where cloud disk is to be created.
- zoneid: the zone where cloud disk is to be created.

Create a service

```
 kind :  Persistent  VolumeClai  m
 apiVersion :  v1
 metadata :
   name :  disk - common
 spec :
   accessMode  s :
    -  ReadWriteO  nce
   storageCla  ssName :  alicloud - disk - common - hangzhou - b
   resources :
     requests :
       storage :  20Gi
---
 kind :  Pod
 apiVersion :  v1
 metadata :
   name :  disk - pod - common
 spec :
   containers :
  -  name :  disk - pod
     image :  nginx
     volumeMoun  ts :
      -  name :  disk - pvc
         mountPath : "/ mnt "
   restartPol  icy : " Never "
   volumes :
    -  name :  disk - pvc
       persistent  VolumeClai  m :
         claimName :  disk - common
```

Default options

By default, the cluster provides the following StorageClasses, which can be used in a single AZ cluster.

- alicloud-disk-common: basic cloud disk.
- alicloud-disk-efficiency: high-efficiency cloud disk.
- alicloud-disk-ssd: SSD disk.
- alicloud-disk-available: provides highly available options, first attempts to create a high-efficiency cloud disk. If the corresponding AZ's efficient cloud disk resources

are sold out, tries to create an SSD disk. If the SSD is sold out, tries to create a common cloud disk.

Creating a multi-instance StatefulSet using cloud disk

Use volume Claim Templates that dynamically creates multiple PVCs and PVs and binds them.

```
 apiVersion :  v1
 kind :  Service
 metadata :
   name :  nginx
   labels :
     app :  nginx
 spec :
   ports :
  - port :  80
     name :  web
   clusterIP :  None
   selector :
     app :  nginx
---
 apiVersion :  apps / v1beta2
 kind :  StatefulSe  t
 metadata :
   name :  web
 spec :
   selector :
     matchLabel  s :
       app :  nginx
   serviceNam  e : " nginx "
   replicas :  2
   template :
     metadata :
       labels :
         app :  nginx
     spec :
       containers :
       -  name :  nginx
          image :  nginx
          ports :
         -  containerP  ort :  80
            name :  web
          volumeMoun  ts :
         -  name :  disk - common
            mountPath : / data
   volumeClai  mTemplates :
  -  metadata :
       name :  disk - common
     spec :
       accessMode  s : [ " ReadWriteO  nce " ]
       storageCla  ssName : " alicloud - disk - common "
       resources :
         requests :
```

```
            storage :  10Gi
```

1.10.2 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 "
      volumeMoun  ts :
       -  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.

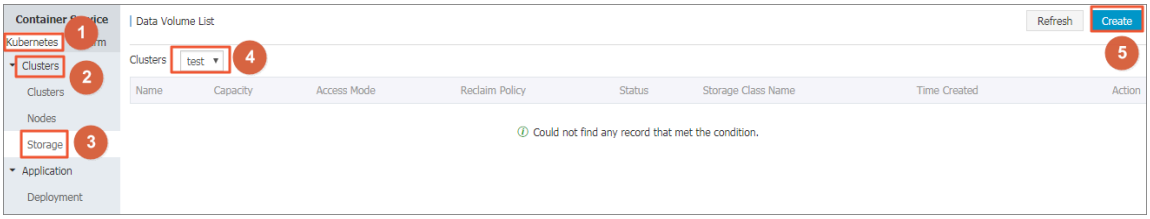
```
apiVersion :  v1
kind :  Persistent  Volume
metadata :
  name :  pv - nas
spec :
  capacity :
    storage :  5Gi
  storageCla  ssName :  nas
  accessMode  s :
   -  ReadWriteM  any
```

```
   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, clickCluster > Storagein 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.

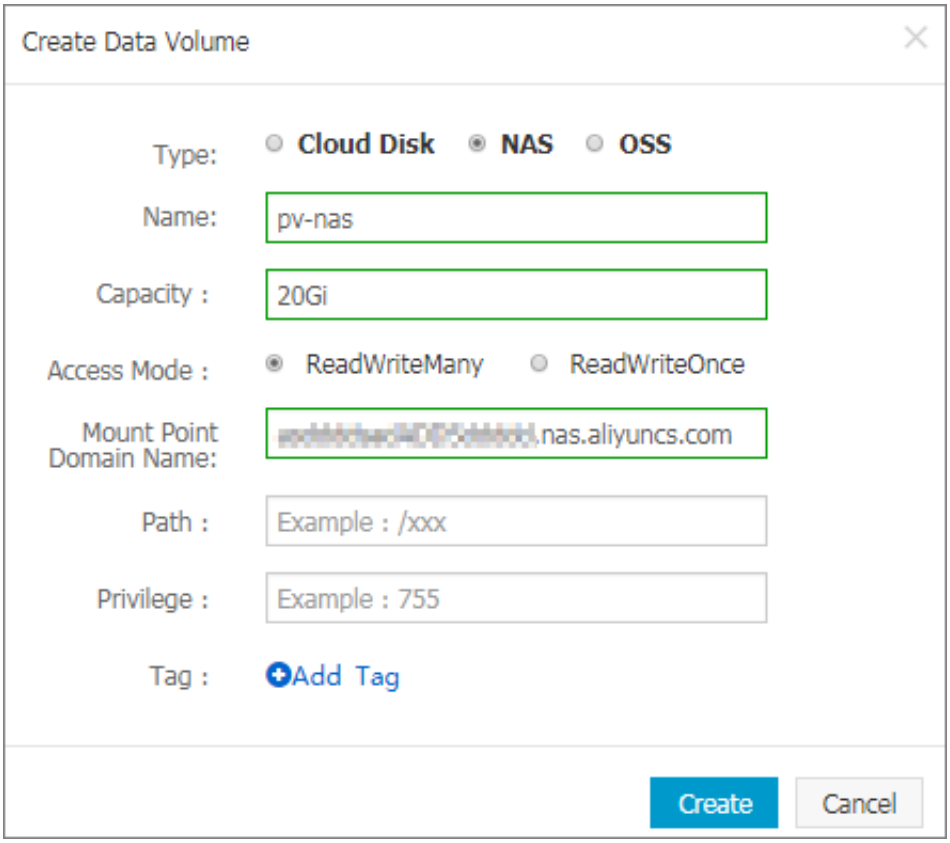
     ■ If this field is left empty, the data volume is mounted to the NAS root directory by default.

   - 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 "
      volumeMoun  ts :
         -  name :  pvc - nas
            mountPath : "/ data "
  volumes :
 -  name :  pvc - nas
    persistent  VolumeClai  m :
        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 :  Persistent  Volume
  metadata :
     name :  nas
  spec :
     capacity :
       storage :  8Gi
     accessMode  s :
      -  ReadWriteM  any
     persistent  VolumeRecl  aimPolicy :  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 :  Persistent  VolumeClai  m
metadata :
     name :  nasclaim
spec :
     accessMode  s :
      -  ReadWriteM  any
     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
         volumeMoun  ts :
        -  mountPath : "/ var / www / html "
           name :  mypd
     volumes :
      -  name :  mypd
         persistent  VolumeClai  m :
           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 :  StorageCla  ss
metadata :
  name :  alicloud - nas
provisione  r :  alicloud / nas
---
apiVersion :  v1
kind :  ServiceAcc  ount
metadata :
```

```
    name :  alicloud - nas - controller
    namespace :  kube - system
---
 kind :  ClusterRol  eBinding
 apiVersion :  rbac . authorizat  ion . k8s . io / v1beta1
 metadata :
    name :  run - alicloud - nas - controller
 subjects :
  -  kind :  ServiceAcc  ount
     name :  alicloud - nas - controller
     namespace :  kube - system
 roleRef :
    kind :  ClusterRol  e
    name :  alicloud - disk - controller - runner
    apiGroup :  rbac . authorizat  ion . 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 :
        toleration  s :
       -  effect :  NoSchedule
          operator :  Exists
          key :  node - role . kubernetes . io / master
       -  effect :  NoSchedule
          operator :  Exists
          key :  node . cloudprovi  der . kubernetes . io / uninitiali
 zed
        nodeSelect  or :
           node - role . kubernetes . io / master : ""
        serviceAcc  ount :  alicloud - nas - controller
        containers :
         -  name :  alicloud - nas - controller
            image :  registry . cn - hangzhou . aliyuncs . com / acs /
 alicloud - nas - controller : v1 . 8 . 4
            volumeMoun  ts :
           -  mountPath : / persistent  volumes
              name :  nfs - client - root
            env :
             -  name :  PROVISIONE  R_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 :  StatefulSe  t
metadata :
  name :  web
spec :
  serviceNam  e : " nginx "
  replicas :  2
  volumeClai  mTemplates :
 -  metadata :
      name :  html
    spec :
      accessMode  s :
       -  ReadWriteO  nce
      storageCla  ssName :  alicloud - nas
      resources :
        requests :
          storage :  2Gi
  template :
    metadata :
      labels :
        app :  nginx
    spec :
      containers :
     -  name :  nginx
        image :  nginx : alpine
        volumeMoun  ts :
       -  mountPath : "/ usr / share / nginx / html /"
          name :  html
```

1.10.3 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.

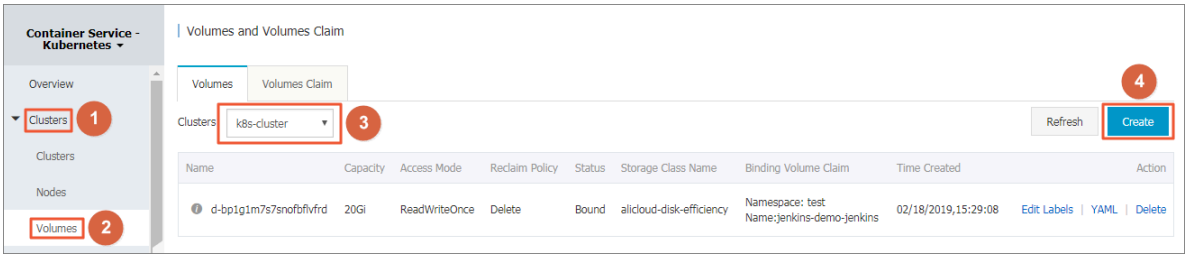
Create PV by using YAML file

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

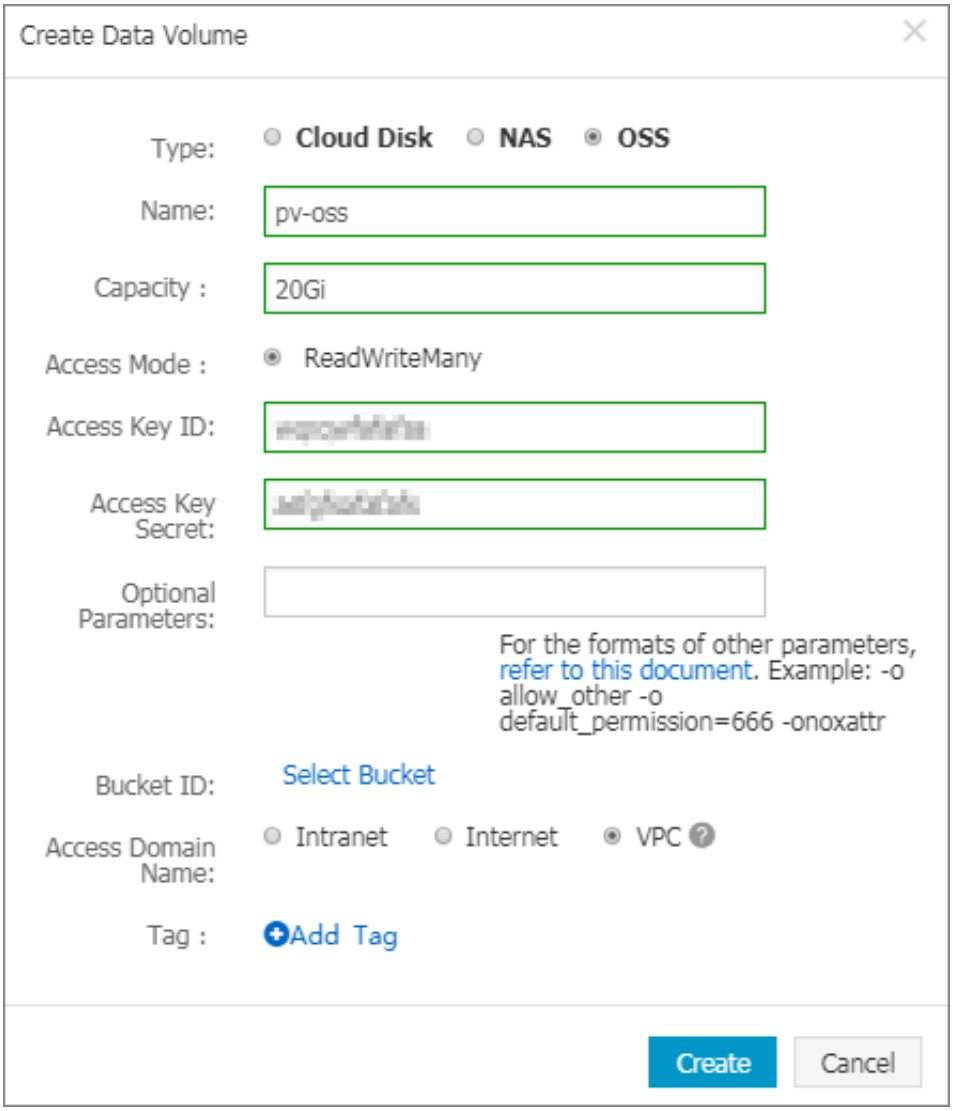
1. Log on to the *Container Service console*.
2. In the left-side navigation pane under Kubernetes, choose Clusters > Storage.
3. Select the cluster from the Clusters drop-down list and then click Create in the upper-right corner.



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 "
      volumeMoun  ts :
         -  name :  pvc - oss
            mountPath : "/ data "
  volumes :
 -  name :  pvc - oss
    persistent  VolumeClai  m :
        claimName :  pvc - oss
```

Use OSS dynamic storage volumes

Currently not supported.

1.11 Storage claim management

1.11.1 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*.

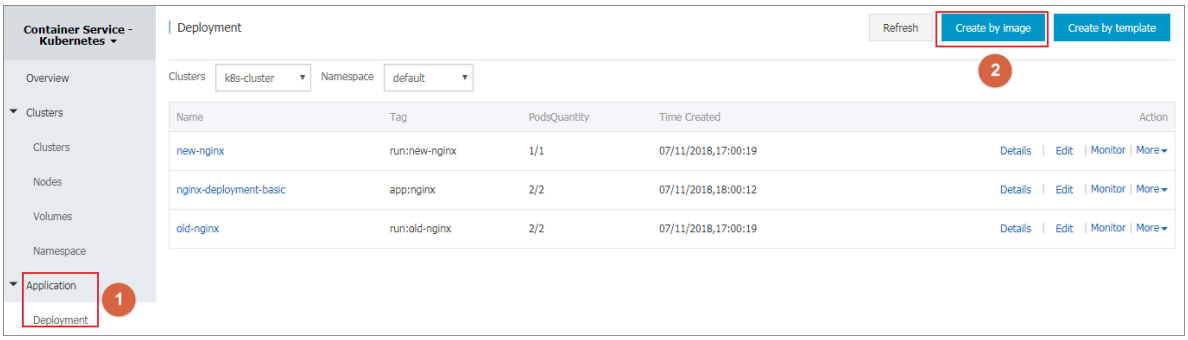
Prerequisites

- **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*.

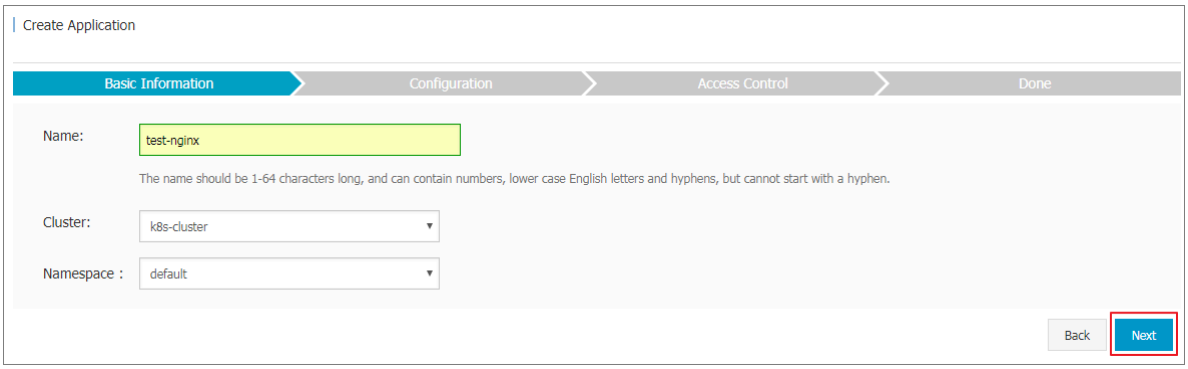
Procedure

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

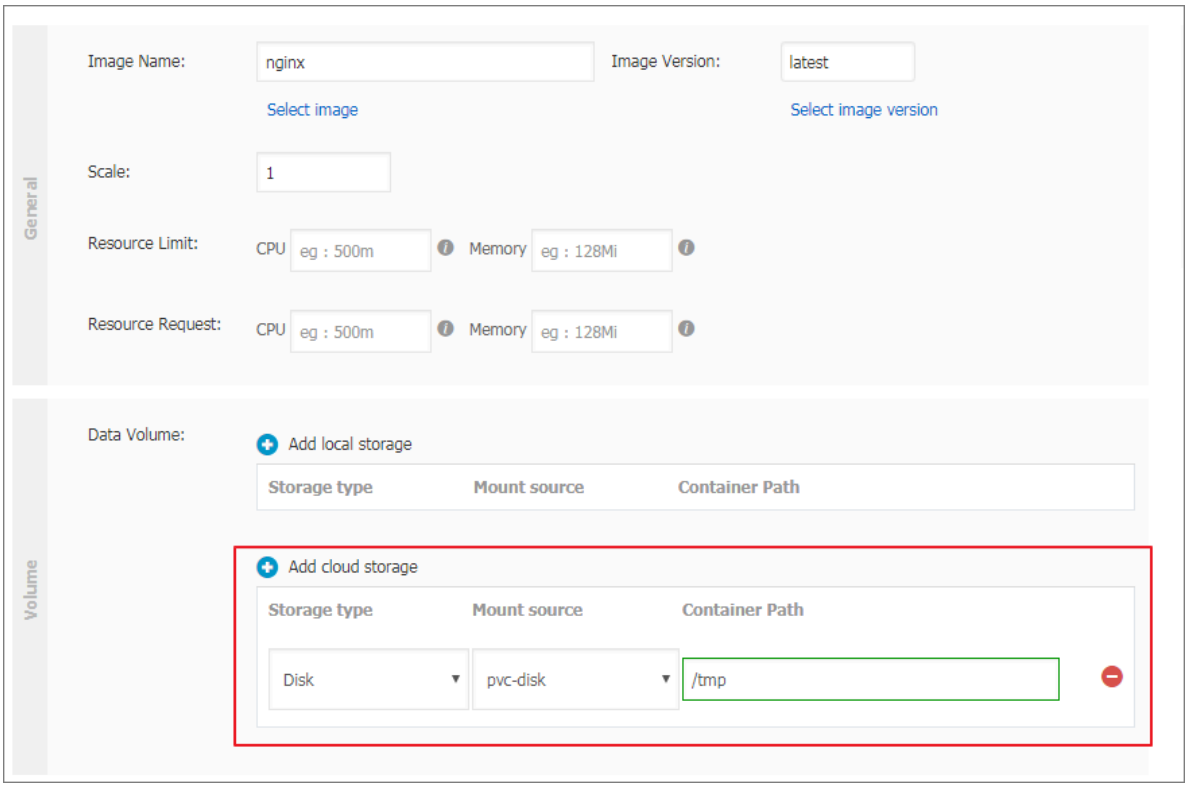
2. Under Kubernetes, click Application > Deployment in the left-side navigation pane. Enter the Deployment List page and click Create by image in the upper-right corner.



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

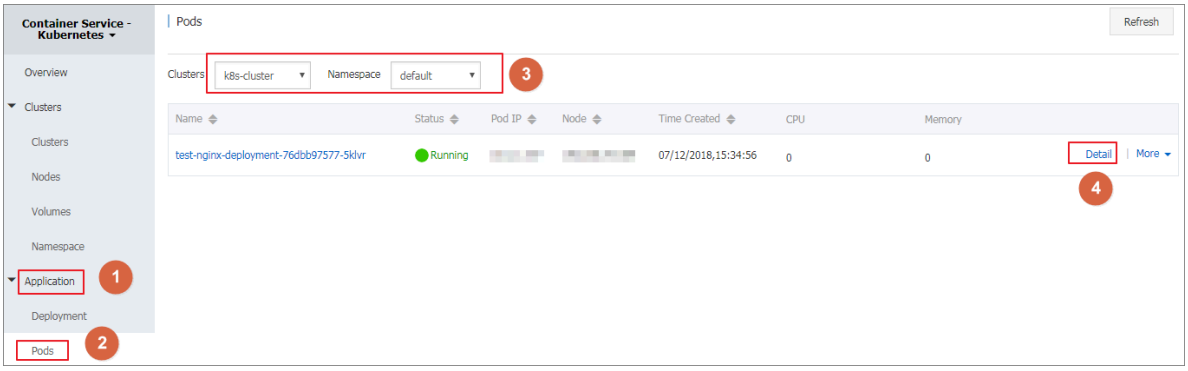


4. On the Application Configuration page, select Image. Then configure the cloud storage type of data volume, cloud disk, NAS, and OSS types are supported. In this example, use the cloud storage volume claim and click Next.

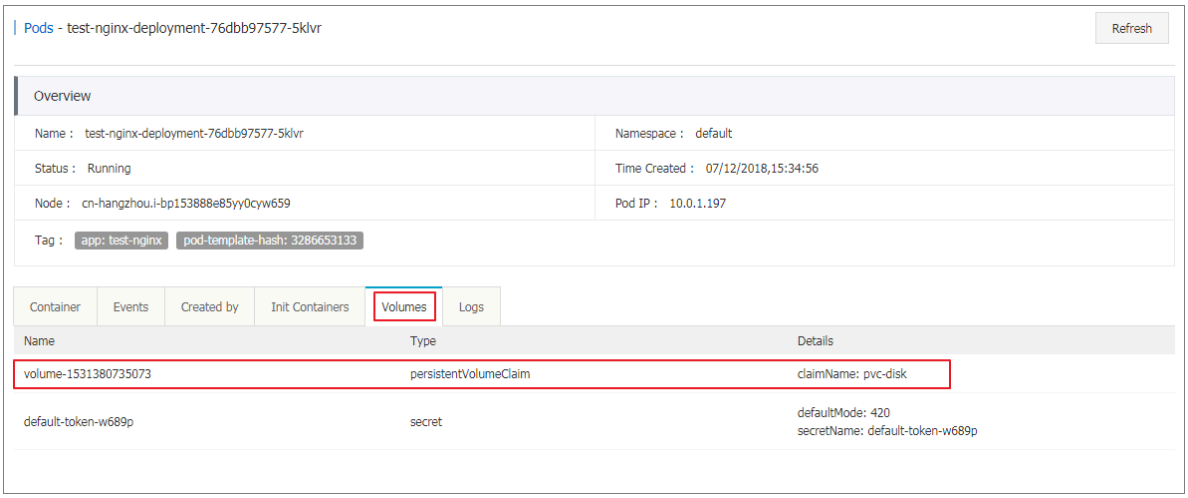


5. See *Create a service* to configure the test-nginx application, and click Create.

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.



1.12 Logs

1.12.1 Application log management

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

- Following the instructions of *#unique_43*, you can make the best use of the functions provided by Alibaba Cloud Log Service, such as log statistics and analysis.
- With *Log-pilot*, an open source project provided by Alibaba Cloud Container Service, and *#unique_44*, you can easily build your own application log clusters.

1.12.2 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.

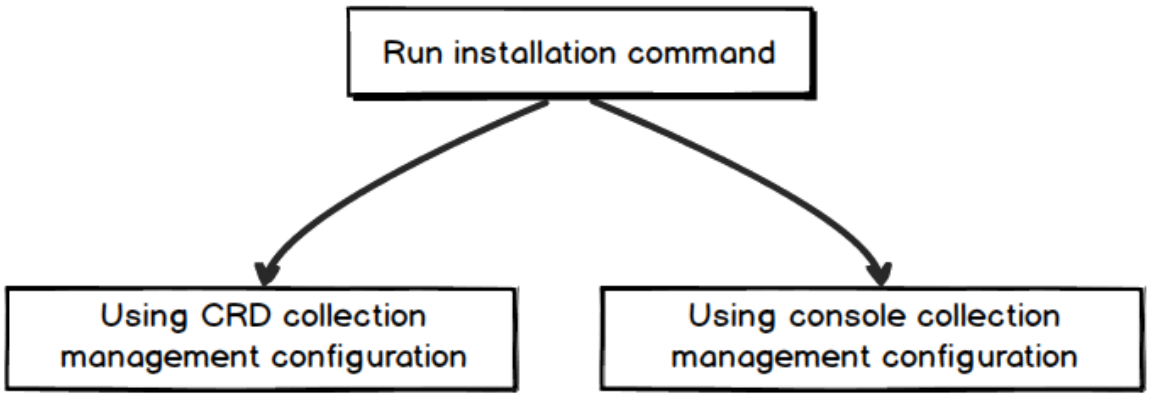
Collection procedure

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_c luster_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_c  luster_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 / ClusterRol  eBinding
 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   ContainerC  reating   0   0s
 logtail - q5s87   0 / 1   ContainerC  reating   0   0s
 alibaba - log - controller - 7cf6d7dbb5 - qvn6w   0 / 1   ContainerC
 reating   0   0s


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


==>  v1beta1 / CustomReso  urceDefini  tion
 NAME   AGE
 aliyunlogc  onfigs . log . alibabaclo  ud . com   0s


==>  v1beta1 / ClusterRol  e
 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:

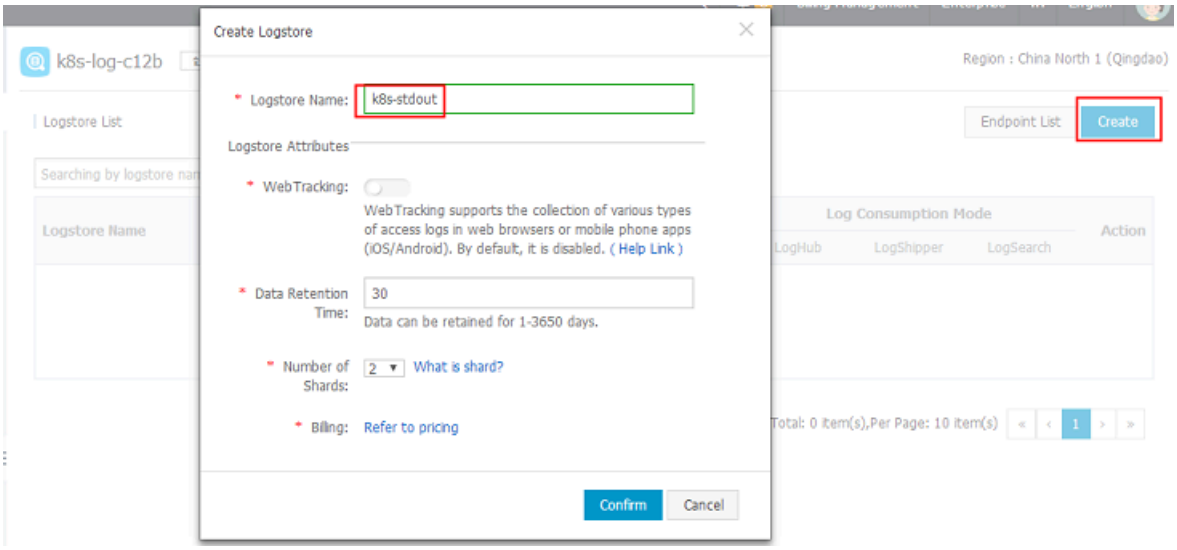
1. Go to the Logstore List page.

   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



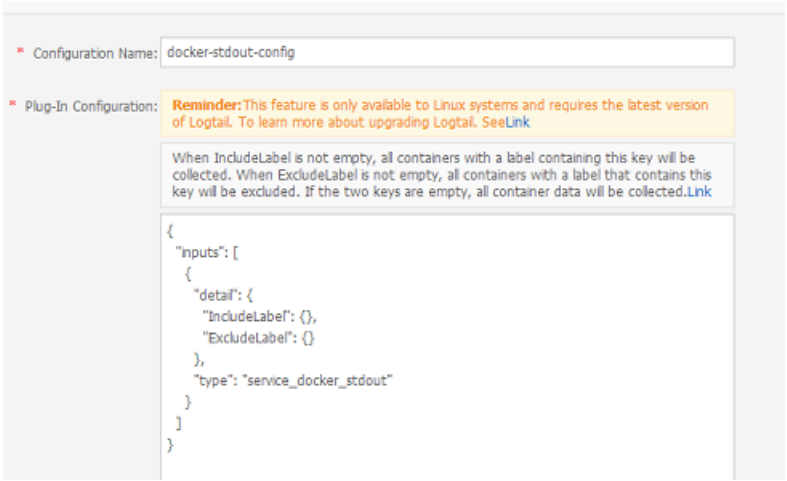
3. Configure the collection.

   a. Go to the Data Import Wizard page.

   b. 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.

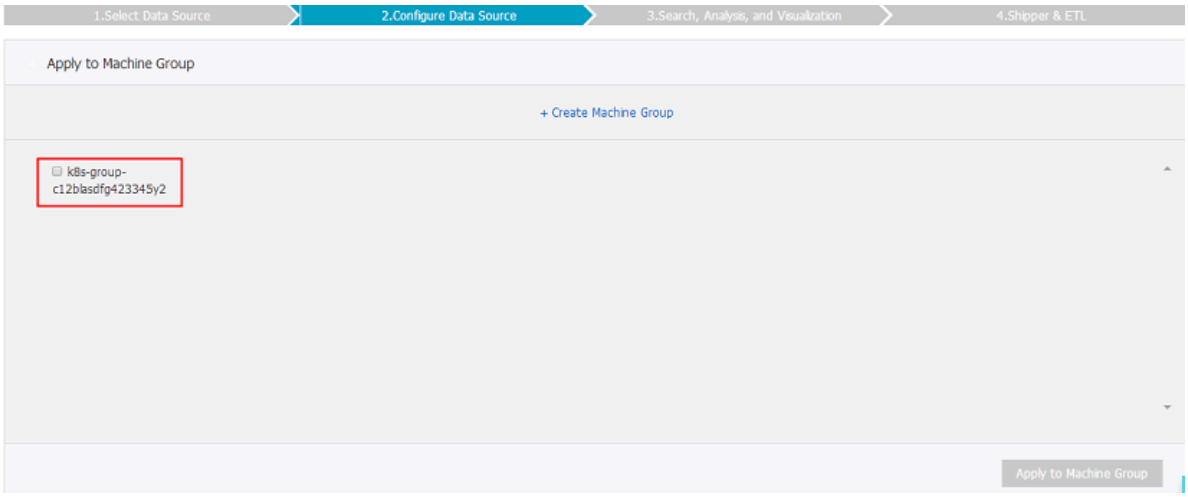
      Figure 1-3: Docker stdout



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

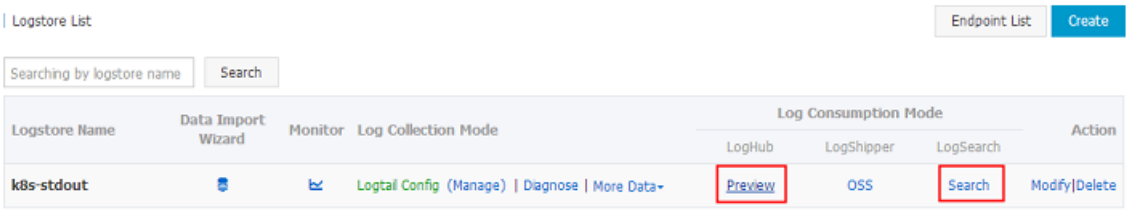


Now you have configured the collection. To configure indexes and log shipping, continue with the follow-up configurations. You can also exit the current page to complete the configuration.

View collected logs

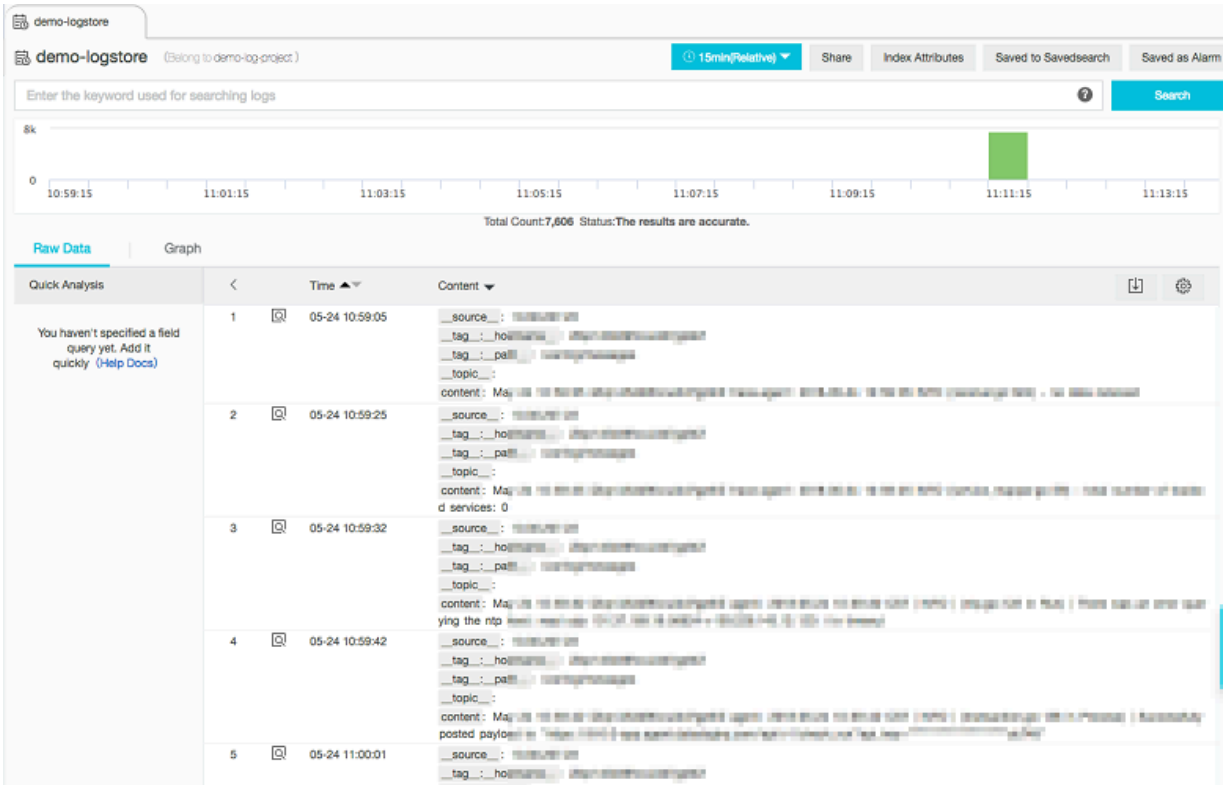
Based on the collection configuration, Logtail can collect stdout logs one minute after a container in your cluster receives stdout input. On the Logstore List page, click Preview to quickly preview collected logs, or click Search to customize searching and analysis of these logs.

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*.

1.12.3 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_log_s_$name = $path`. This environment variable contains the following two variables:

- One variable is $name, a custom string which indicates different meanings in different scenarios. In this scenario, $name indicates index when collecting logs to Elasticsearch.
- The other is $path which supports two input modes, stdout and paths of log files within containers, respectively corresponding to the standard output of logs and log files within containers.
  - Stdout indicates to collect standard output logs from containers. In this example, to collect Tomcat container logs, configure the label `aliyun.logs.catalina = stdout` to collect standard output logs of Tomcat.
  - The path of a log file within a container also supports wildcards. To collect logs within the Tomcat container, configure the environment variable `aliyun_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 "`.

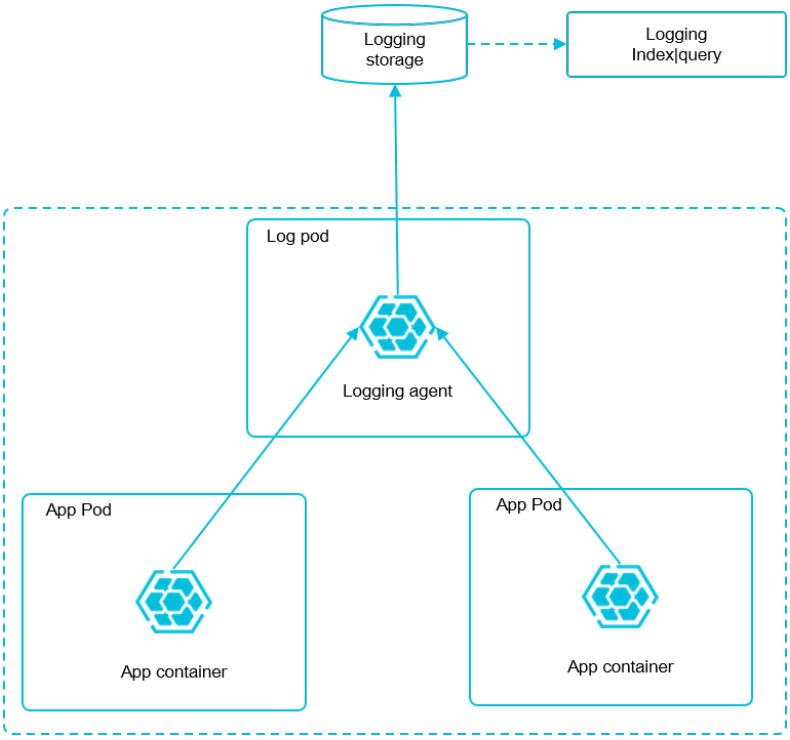
Besides, log-pilot supports multiple log parsing formats, including none, JSON, CSV, Nginx, apache2, and regxp. You can use the `aliyun_logs_$name_format =<format>` label to tell log-pilot to use what format to parse logs when collecting logs.

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
statefulset.apps/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_log s_catalina

      value : " stdout " ## Collect standard output logs .
    - name : aliyun_log s_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_log s_catalina = stdout` indicates to collect stdout logs from the container.
- `aliyun_log s_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:

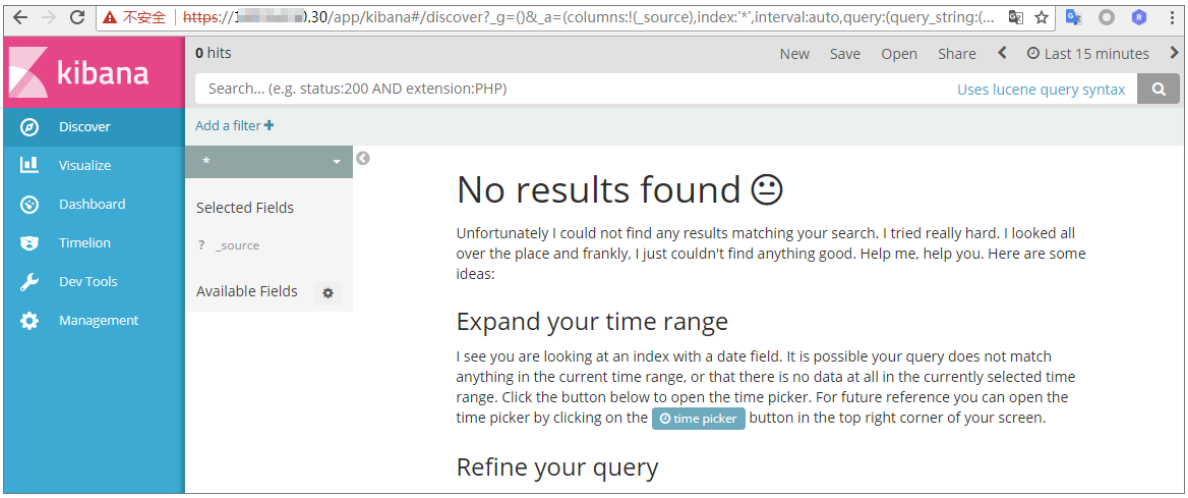
```
apiVersion :  extensions / v1beta1
 kind :  Ingress
 metadata :
   name :  kibana - ingress
   namespace :  kube - system  # Make  sure  the  namespace  is
  the  same  as  that  of  the  Kibana  service .
 spec :
   rules :
  -  http :
       paths :
      -  path : /
         backend :
```

```
            serviceNam  e :  kibana  # Enter  the  name  of  the
   Kibana   service .
            servicePor  t :  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.



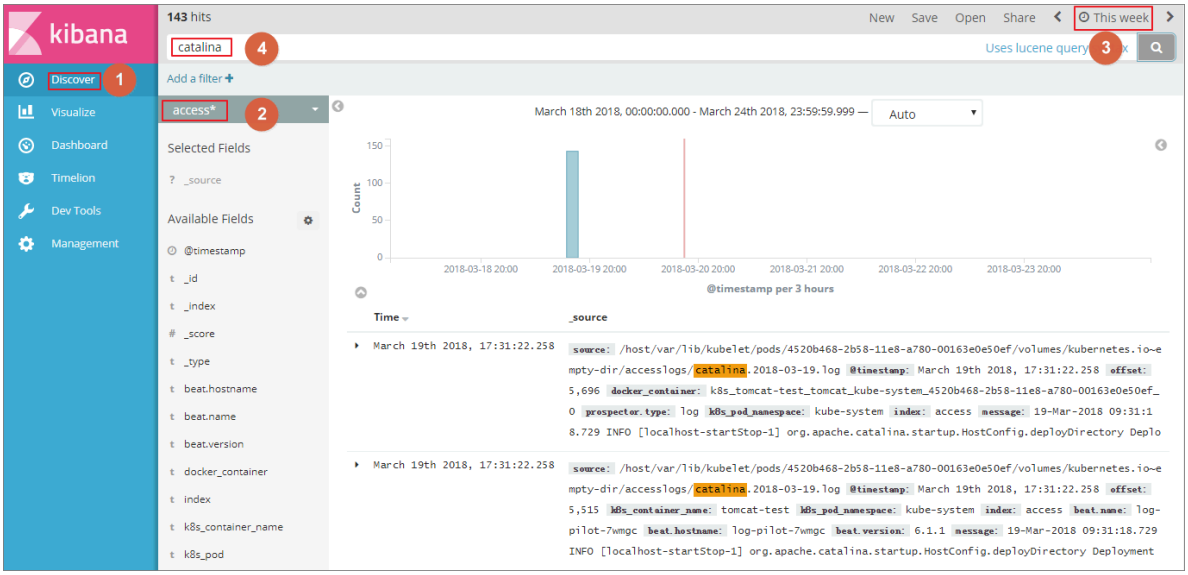
4. Click Management in the left-side navigation pane. Then, click Index Patterns > Create Index Pattern. The detailed index name is the `$ name` variable suffixed with a time string. You can create an index pattern by using the wildcard `*`. In this example, use `$ name *` to create an index pattern.

   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
 correspond  ing  pod  of  Elasticsea  rch .
   ...
$  kubectl  exec  - it  elasticsea  rch - 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  x06jj19PS4  Cim6Ajo51P  Wg  1  1  4
 0  53 . 6kb  26 . 8kb
  green  open  access - 2018 . 03 . 19  txd3tG - NR6 -
 guqmMEKKzE  w  5  1  143  0  823 . 5kb  411 . 7kb
```

```
  green   open   catalina - 2018 . 03 . 19   ZgtWd16FQ7   qqJNNWXxFP
cQ   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.

1.13 FAQ

1.13.1 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   descriptio  n   are
   the   descriptio  ns   of   pod   running   status .  You   can
 analyze   pod   errors   based   on   the   descriptio  ns .
```

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 - bp19myla3u  vnt6zihejb / 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   descriptio  n   are
   the   descriptio  ns   of   pod   running   status .  You   can
 analyze   pod   errors   based   on   the   descriptio  ns .
```

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 - bp19myla3u  vnt6zihejb / 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.

1.13.2 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*.

1.13.3 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:"a80231648a1473929271764b920a8e346f6de844", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.9.1", GitCommit:"a80231648a1473929271764b920a8e346f6de844", GitTreeState:"clean"}
```

2 Authorizations

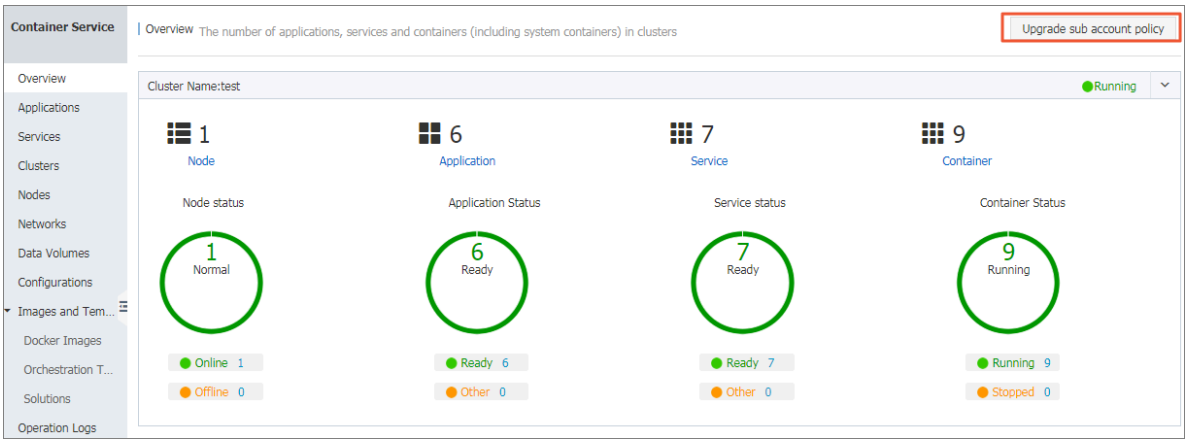
2.1 Upgrade sub-account policy

Container Service comprehensively upgrades the security authorization management on January 15 2018, and provides cross-service authorization based on STS to provide you with more secure services. If you have used Container Service before 15 January 2018, the system completes the authorization by default. For more information about the granted permissions, see *Role authorization* If you used Container Service with a sub-account before, grant the sub-account the permissions to use Container Service again.

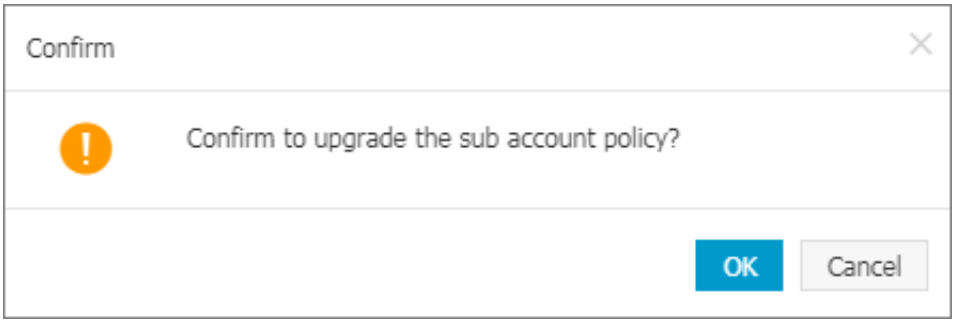
Container Service can automatically upgrade the sub-account policy. With this feature, Container Service automatically grants your sub-accounts the AliyunCSReadOnlyAccess permission. You can also select to manually grant permissions to your sub-accounts in the Resource Access Management (RAM) console.

Upgrade sub-account policy

1. Use the primary account to log on to the *Container Service console*.
2. Click Upgrade sub account policy in the upper-right corner on the Overview page.

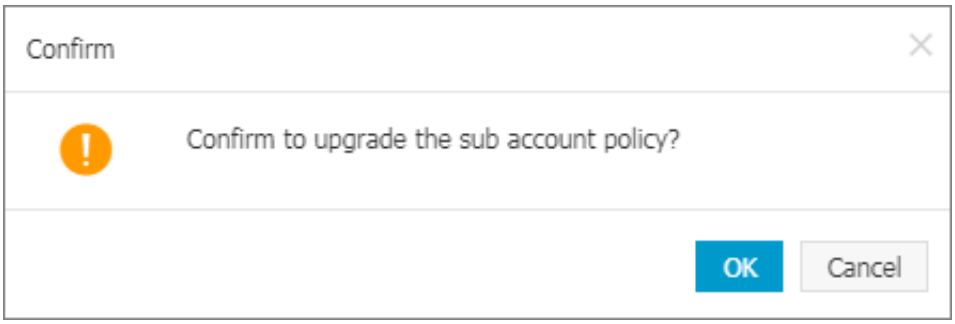


3. Click OK in the displayed dialog box.



Container Service will grant your sub-accounts the corresponding roles when the sub-account policy is being upgraded.

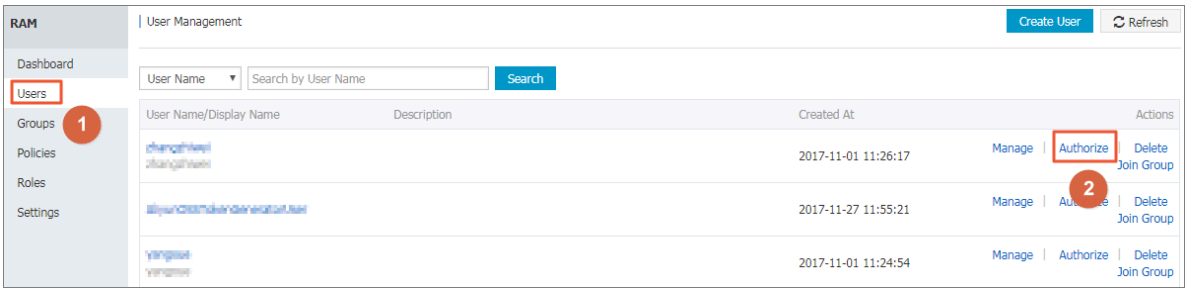
If the upgrade fails, a dialog box listing the sub-accounts that fail to be upgraded appears.



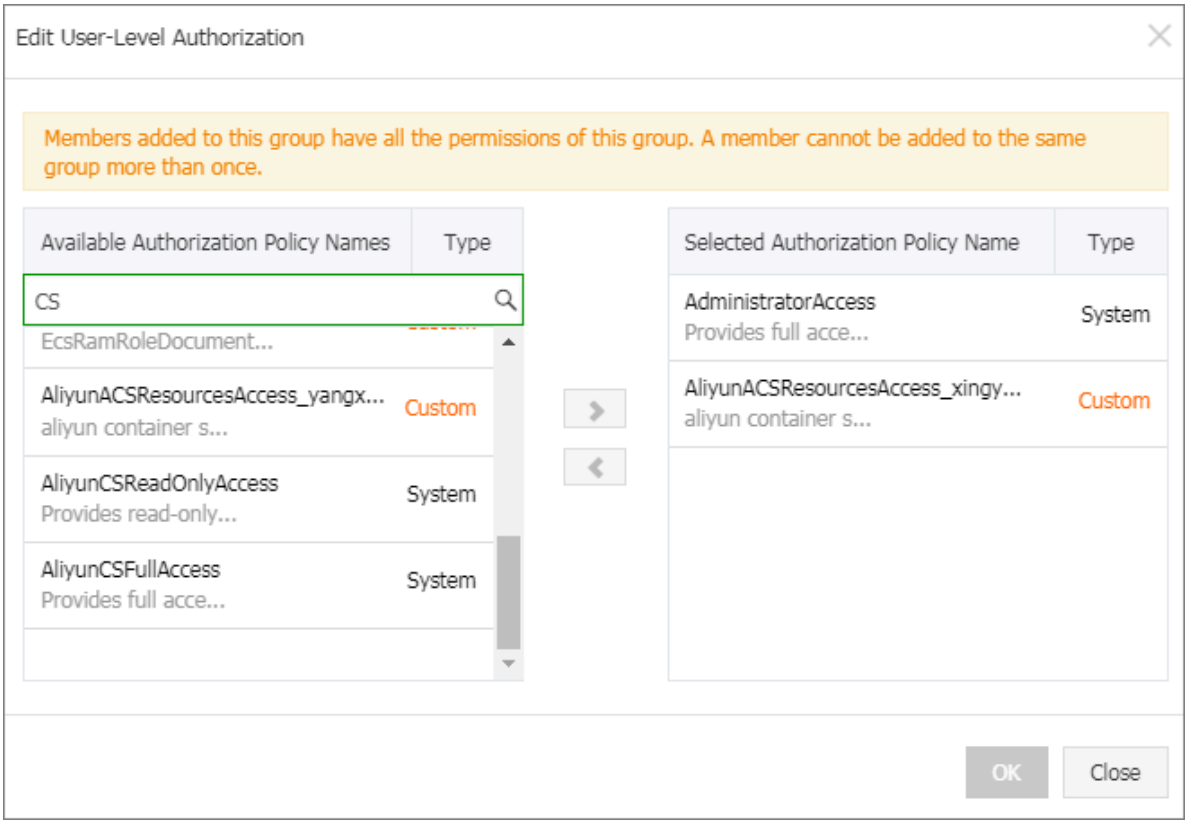
Click Upgrade sub account policy to try to upgrade again or go to the RAM console to manually grant permissions to sub-accounts.

Grant permissions to sub-accounts in RAM console

1. Use the primary account to log on to the *Container Service console*.
2. Click Users in the left-side navigation pane.
3. Click Authorize at the right of the sub-account.



4. Select the authorization policy and click 1 to add the policy to the Selected Authorization Policy Name. Click OK.



Container Service provides two system authorization policies:

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

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

3 Clusters

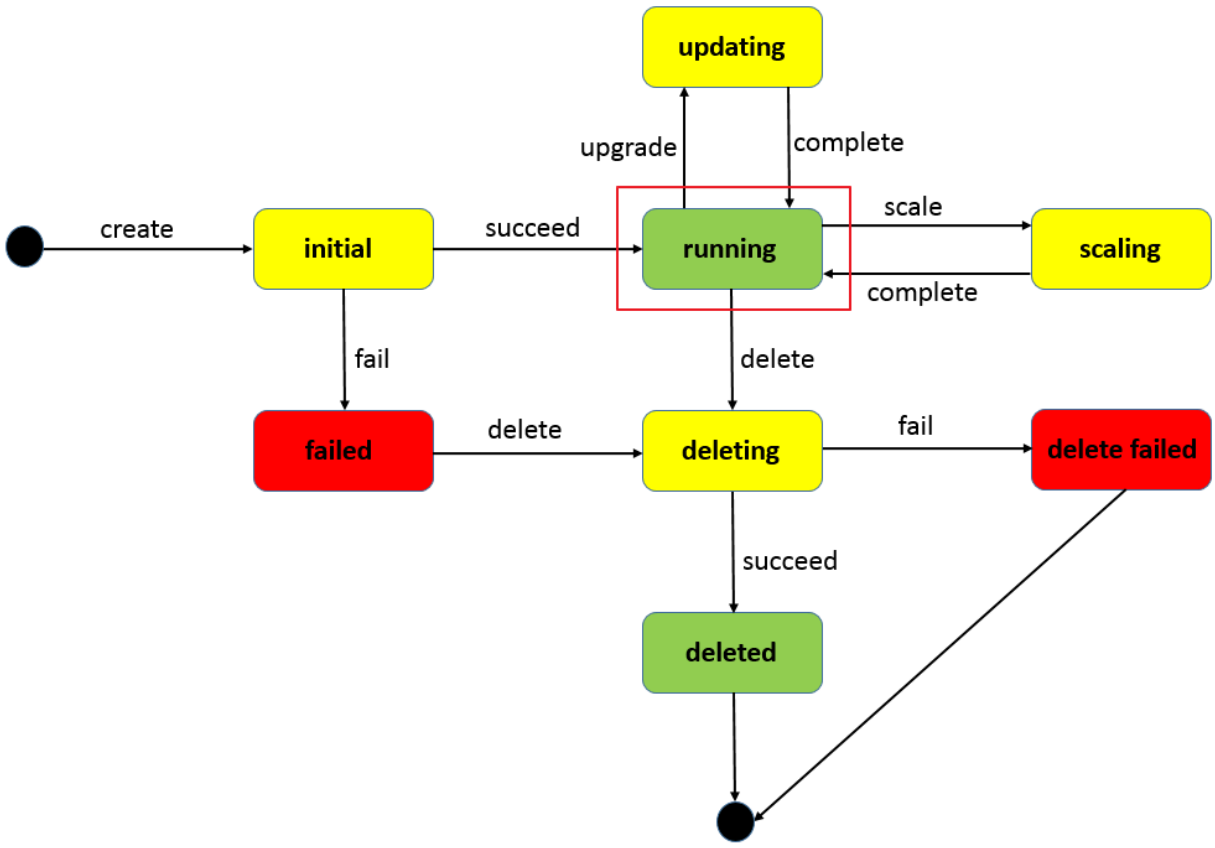
3.1 Cluster lifecycle

Table 3-1: A complete cluster lifecycle includes the following statuses.

Status	Description
inactive	The successfully created cluster does not contain any node.
initial	The cluster is applying for corresponding cloud resources.
running	The cluster successfully applied for the cloud resources.
updating	The cluster is upgrading the Agent.
scaling	Change the number of cluster nodes.
failed	The cluster application for cloud resources failed.
deleting	The cluster is being deleted.
delete_failed	The cluster failed to be deleted.

Status	Description
deleted (invisible to users)	The cluster is successfully deleted.

Figure 3-1: Cluster status flow



3.2 Add an existing ECS instance

You can add a purchased Elastic Compute Service (ECS) instance to a specified cluster.

> **Note:**
> At most 20 ECS instances can be added to a cluster by default. To add more ECS instances, *open a ticket*.

You can add an existing ECS instance in the following ways:

- Add ECS instances automatically: The image and system disk of the ECS instance are reset by using this method. You can add one or more ECS instances to the cluster at a time.
- Add the ECS instance manually: Manually add the ECS instance by running scripts on the ECS instance. You can only add one ECS instance to the cluster at a time.

Prerequisites

If you have not created a cluster before, create a cluster first. For information about how to create a cluster, see *Create a cluster*.

Instructions

- The ECS instance to be added must be in the same region and use the same network type (Virtual Private Cloud (VPC)) as the cluster.
- When adding an existing ECS instance, make sure that your ECS instance has an Elastic IP (EIP) for the network type VPC, or the corresponding VPC has configured 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.
- If you select to manually add the ECS instance, note that:
  - If you have already installed Docker on your ECS instance, the ECS instance may fail to be added. We recommend that you uninstall Docker and remove the Docker folders before adding the ECS instance by running the following command:

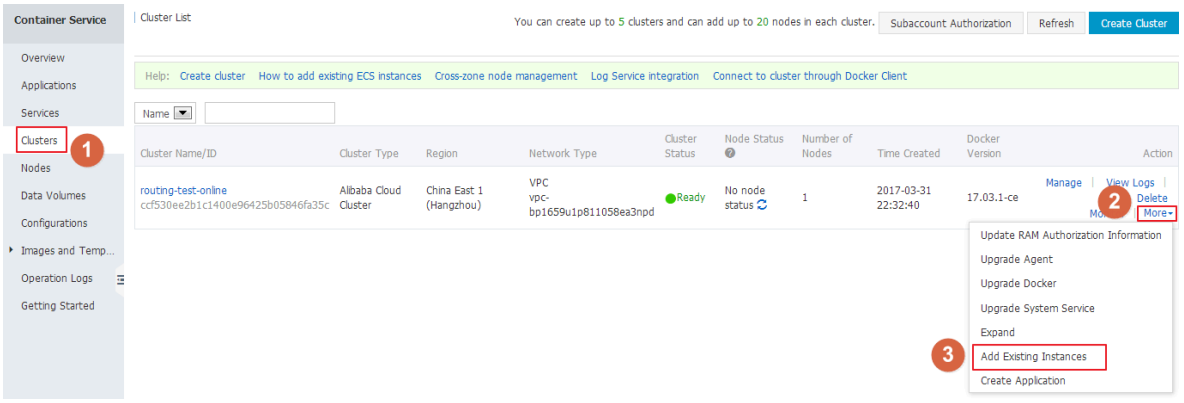
    Ubuntu: `apt - get remove - y docker - engine , rm - fr / etc / docker / / var / lib / docker / etc / default / docker`

    CentOS: `yum remove - y docker - engine , rm - fr / etc / docker / var / lib / docker`
  - Container Service nodes have special requirements for the operating system of the ECS instance. We recommend that you use Ubuntu 14.04/16.04 or CentOS 7 as the operating system. We have strictly tested the stability and compatibility of these operating systems.

Procedure

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

3. Click More at the right of the cluster that you want to add ECS instances and then select Add Existing Instances from the drop-down list.



4. Add ECS instances.

   The ECS instances displayed are filtered and synchronized from your ECS instance list according to the region and network type defined by the cluster.

   Add the ECS instances in the following ways:

   - Add ECS instances automatically.

     Note:
     As this method will reset the image and system disk of the ECS instance, proceed with caution. Create a snapshot to back up your data before adding

the ECS instance. For information about how to create a snapshot, see *Create a snapshot*.

a. Select the ECS instances you want to add to the cluster and click Next Step.

   You can add one or more ECS instances at a time.

b. Configure the instance information. Click Next Step and then click Confirm in the confirmation dialog box.

c. Click Finish.

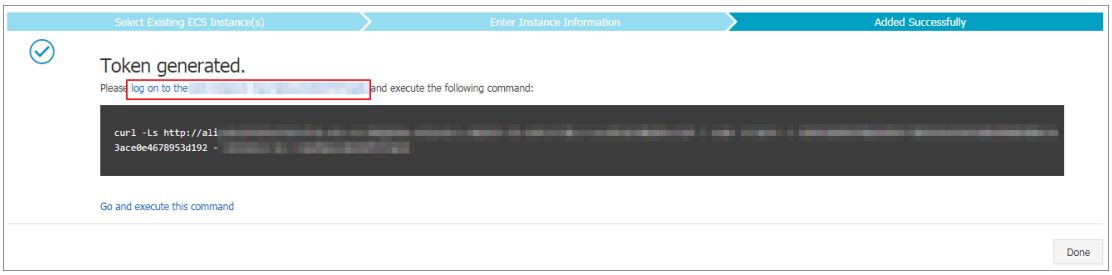
- Manually add the ECS instance by running scripts on the ECS instance.

a. Select Manually Add. Select an ECS instance, and then click Next Step.

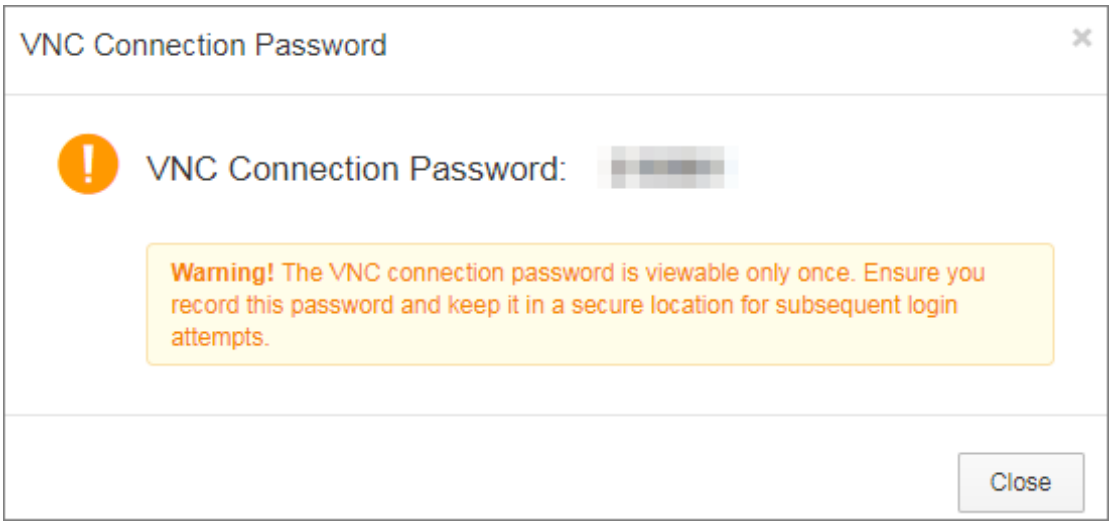
   You can only add one ECS instance at a time.

b. Confirm the instance information and click Next Step.

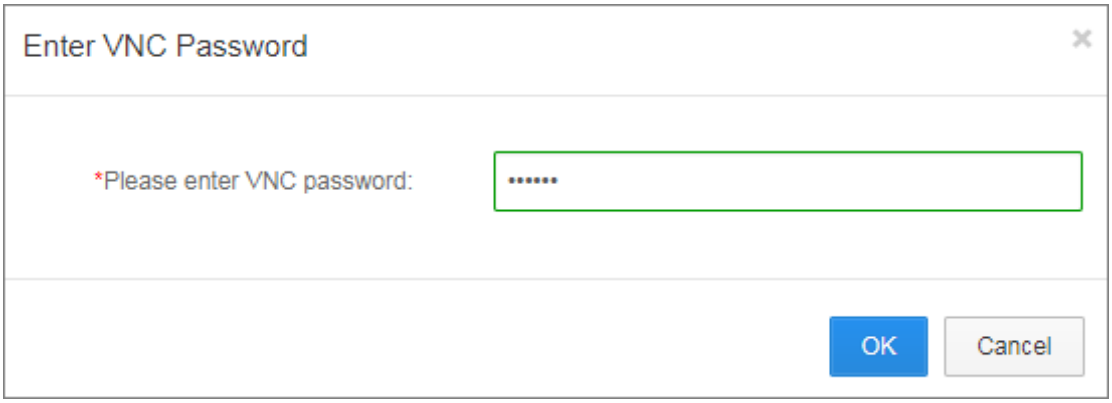
c. The scripts unique to this ECS instance are displayed. Click log on to the ECS instance xxxxxx.



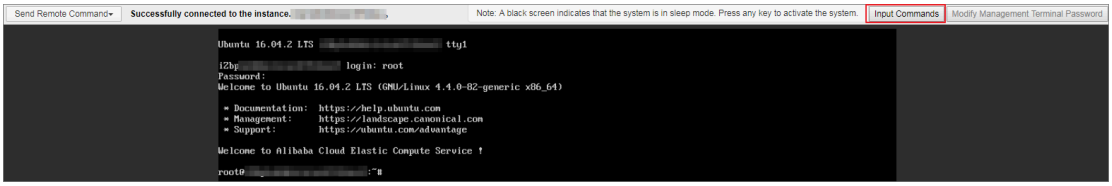
d. The VNC connection password is displayed in the dialog box. Copy the password and click Close.



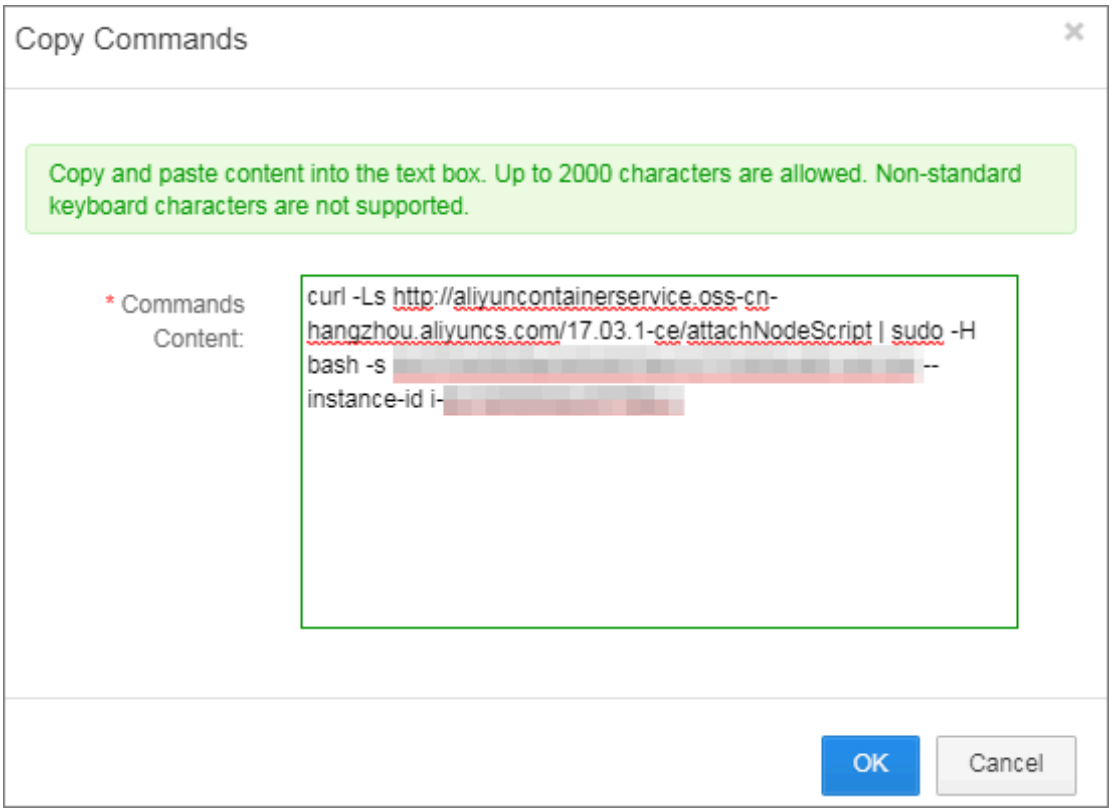
e. In the dialog box, enter the VNC connection password and click OK.



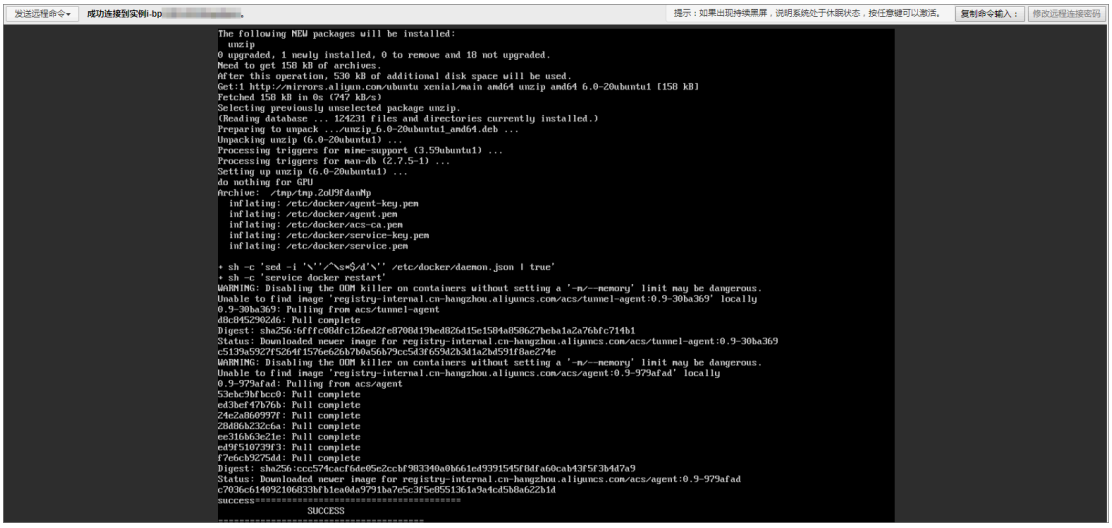
f. Enter the logon account (root) and password of the ECS instance, and press Enter to log on to the ECS instance.



g. Click Input Commands. Paste the preceding scripts into the dialog box, click OK and press Enter.

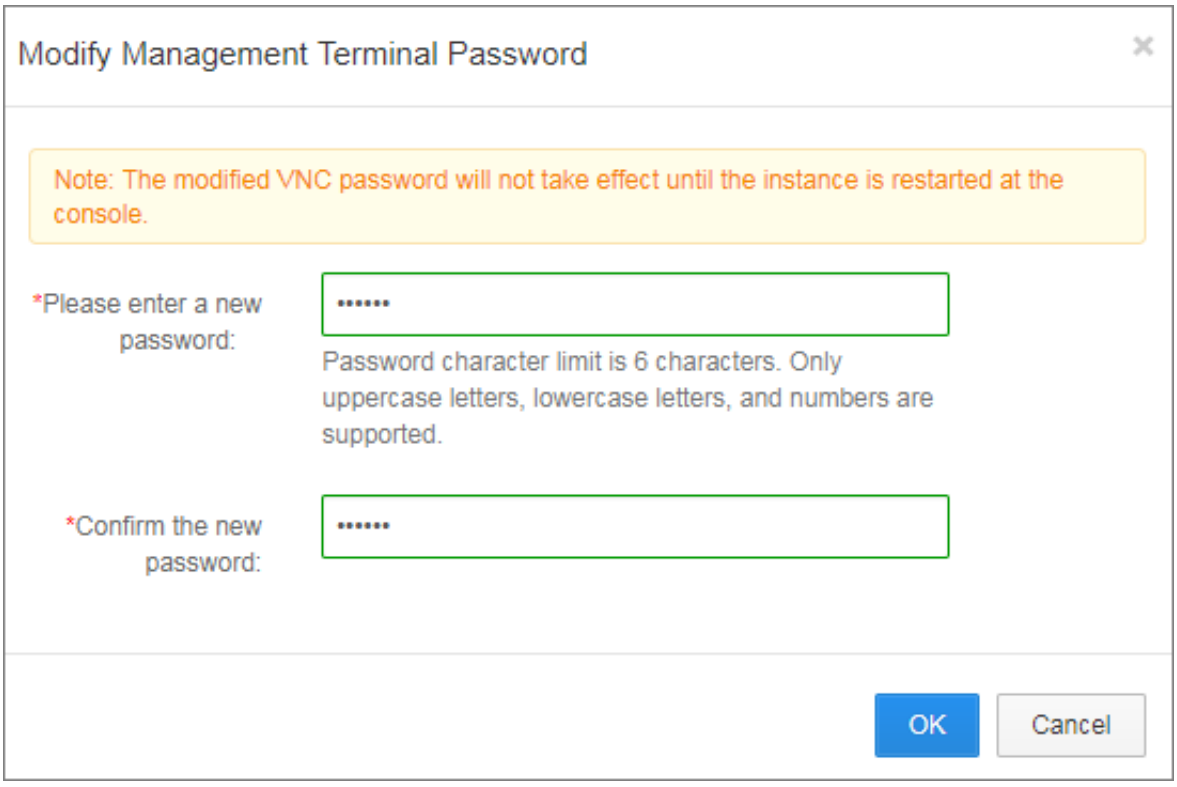


The system runs the scripts. Wait until the scripts are successfully run. A success message is displayed. The ECS instance is successfully added.



Related operation

You can modify the VNC connection password of the ECS instance in the remote terminal connection page. Click Modify Management Terminal Password, enter the new password and click OK in the dialog box.



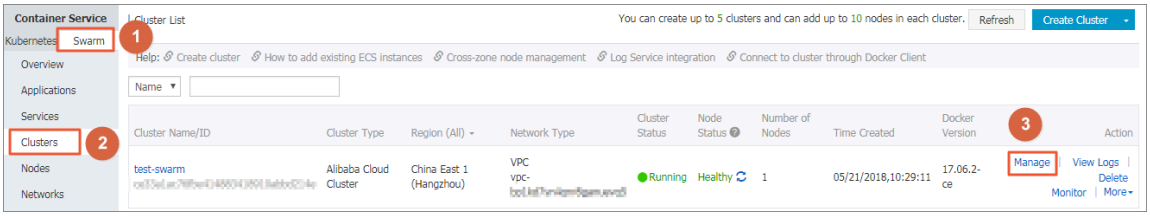
3.3 Download cluster certificate

Context

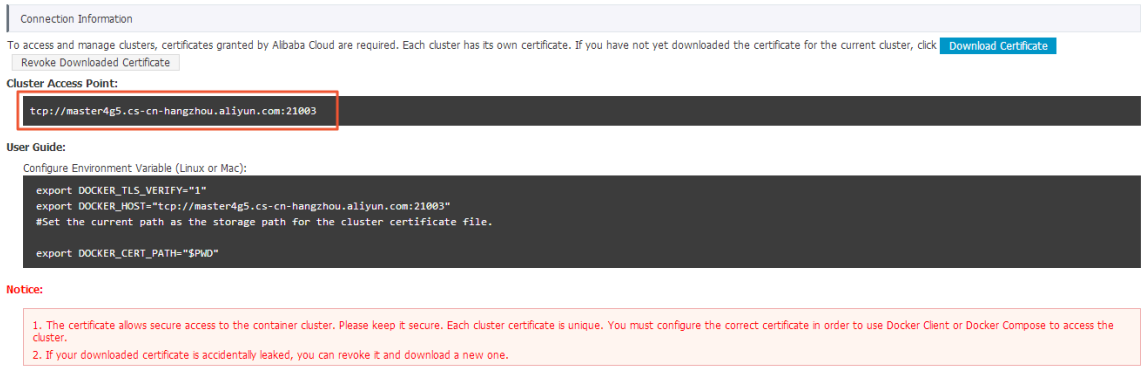
With the downloaded certificate, you can connect to the endpoint exposed from the cluster by using Docker Swarm API or Docker client. For more information, see *Connect to a cluster by using Docker tools*.

Procedure

1. Obtain the access address.
   a) Log on to the *Container Service console*.
   b) Log on to the *Container Service console*.
   c) Click Clusters in the left-side navigation pane. On the Cluster List page, click Manage at the right of a cluster.



   d) The cluster details page is displayed, showing the cluster connection information.



2. Download and save the TLS certificate.

   Configure a TLS certificate before you use the preceding access address to access the Docker cluster.

   Click Download Certificate in the cluster details page to download the TLS certificate. The certFiles.zip file is downloaded. certificate. The `certFiles.zip` file is downloaded. In the following example, the downloaded certificate is saved to the `~/.acs/certs/ClusterName/` directory. `ClusterName` indicates the name of your cluster. You can save the certificate to a different

directory, but we recommend using the *~/.acs/certs/ClusterName/* directory for easy management.

```
mkdir ~/.acs/certs/ClusterName/ # Replace ClusterName with your cluster name
cd ~/.acs/certs/ClusterName/
cp /path/to/certFiles.zip .
unzip certFiles.zip
```

The *certFiles.zip* file contains *ca.pem*, *cert.pem*, and *key.pem*.

3.4 Migrate a cluster

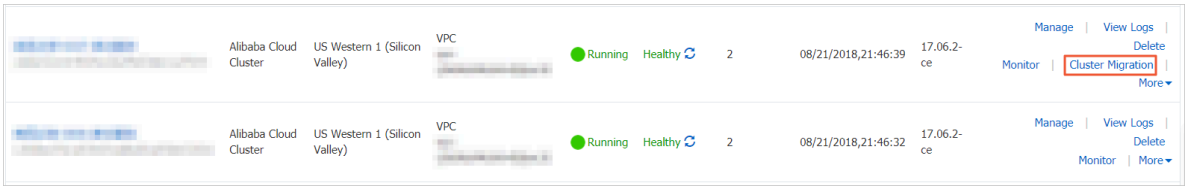
For a Swarm cluster created earlier, you can guarantee the performance and stability of the cluster by migrating the cluster.

Context

- The latest time for migrating a cluster is displayed through SMS, station message, or email. Complete the Swarm cluster migration before the latest time. The system automatically migrates the cluster if you do not migrate the cluster before the latest time.
- Cluster migration rebuilds connections from cluster nodes to the container server without affecting applications deployed in the cluster, nor adding or modifying any data. Make sure that you perform this operation during the low peak period of your business because unpredictable risks might still exist throughout the migration process.

Procedure

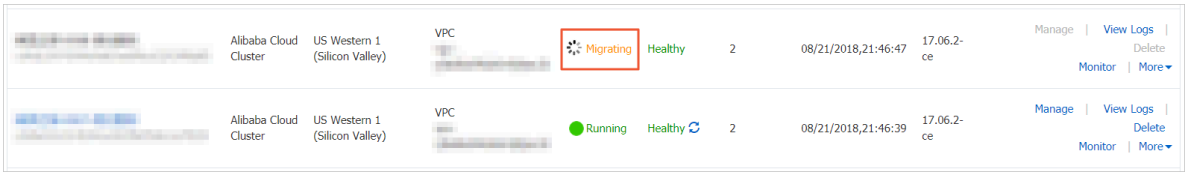
1. Log on to the *Container Service console*.
2. Under the Swarm menu, click Clusters.
3. Click Cluster Migration in the action column at the right of the cluster to be migrated.



4. Click OK in the Prompt dialog box.

> **Note:**
> During cluster migration:
>
> - Information query, deployment, upgrade, and other operations cannot be performed in the console.
> - The cluster cannot be connected to through the cluster access point API.
> - The data and application status in the cluster remain unchanged. Applications deployed on the cluster are still accessible.
> - The migration process takes about three minutes.

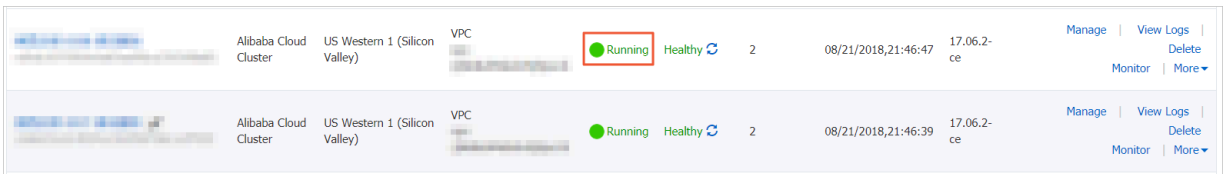
On the Cluster List page, Migrating is displayed in the Cluster Status column.



Result

After cluster migration is completed, on the Cluster List page, Running is displayed in the Cluster Status column.

> **Note:**
>
> - The cluster ID, access point address, and other attributes remain unchanged.
> - Please be sure to confirm that your business is running properly.
> - During the migration process, if you have any questions, please open a ticket in which you include the cluster ID and state whether your deployed applications are normal.



4 Nodes

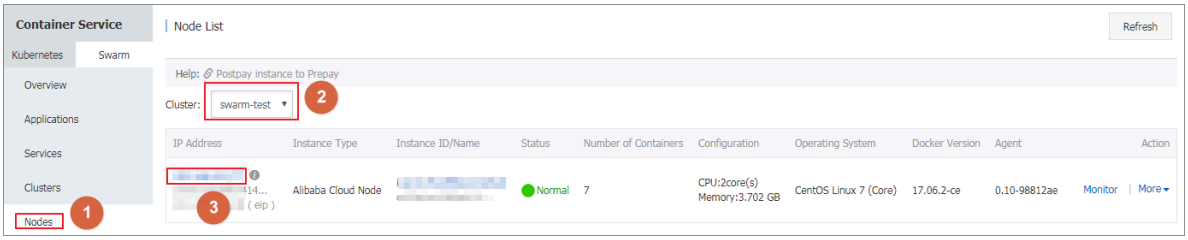
4.1 View containers running on a node

Context

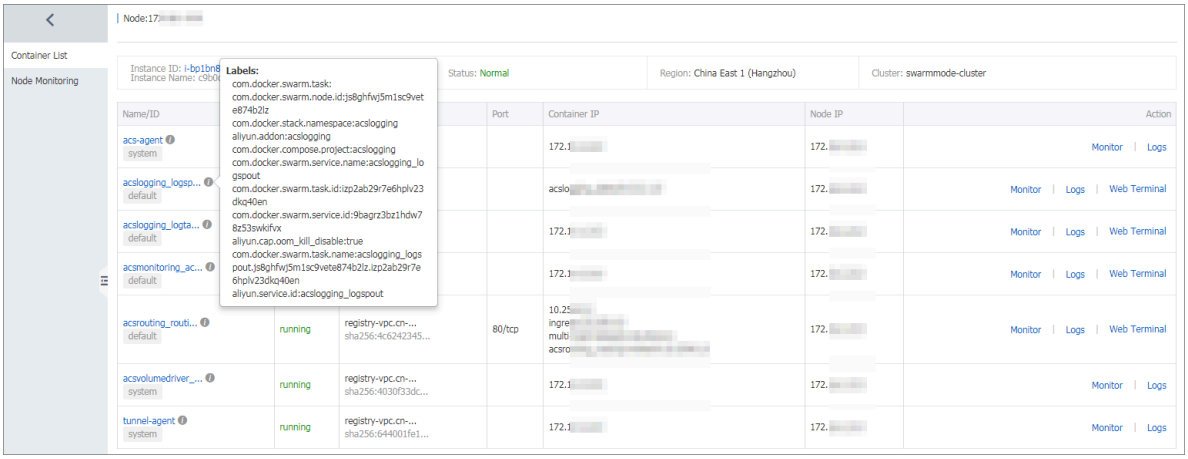
You can view containers running on a node on the Node List page.

Procedure

1. Log on to the *Container Service console*.
2. Click Swarm > Nodes in the left-side navigation pane.
3. On the Node List page, select a cluster from the Cluster drop-down list.
4. Click the node ID.



You can see the list of containers running on the node.



What's next

In the list, you can view the labels, images, the image SHA256 values, logs, and monitoring information of containers and perform operations on containers, including starting and stopping containers, deleting containers, and operating on containers on a remote terminal.

4.2 Update a node certificate

You can update a node certificate of a Swarm cluster to avoid node certificate expiration.

Prerequisites

1. You have created a swarm cluster, see *Create a cluster*.
2. Updating a node certificate reboots the node Docker Daemon. Make sure that containers on the node are all configured to restart automatically.

    Note:
   You can configure a container restart policy when creating an application. When you create an application by using an image, select the Always check box for Restart. When you create an application by using a template, configure a container restart policy in the template `restart : always` .
3. If a node certificate expires within 60 days, a prompt is displayed. You must timely update the node certificate.

Context

Each cluster node has a certificate used to access system control services. Each issued certificate has a valid period. When the valid period of a certificate is about to expire , you must manually renew the certificate. Otherwise, the service of the node is affected.

Procedure

1. Log on to the *Container Service console*.
2. Under the Swarm menu, click Nodes in the left-side navigation pane. The certificate expiration information of each cluster node is displayed.

    Note:
   The certificate expiration time is displayed in the status column only if the node certificate expires within 60 days.
3. Select a node in the node list, and click More > Update Certificate on the right to reissue the node certificate.

    Note:

We recommend that you upgrade the cluster agent to the latest version before updating the node certificate.

4. Optional: If the system prompts you to upgrade the cluster agent after you click Update Certificate, the current cluster agent does not support this feature. You need to upgrade the cluster agent to the new version first, see *Upgrade Agent*. If no prompt is displayed, go to the next step.
5. If no prompt is displayed or the cluster agent is updated, click Update Certificate. Confirm updating information and then update the node cluster certificate.

   Note:

   - When the node certificate update is completed, the Docker Daemon node is automatically restarted about 1 minute later.
   - To guarantee that containers on the node can automatically restart, make sure that an automatic restart policy is configured.
6. After the cluster node certificate is updated, the node certificate information is no longer displayed.

5 Images and templates

5.1 Update an orchestration template

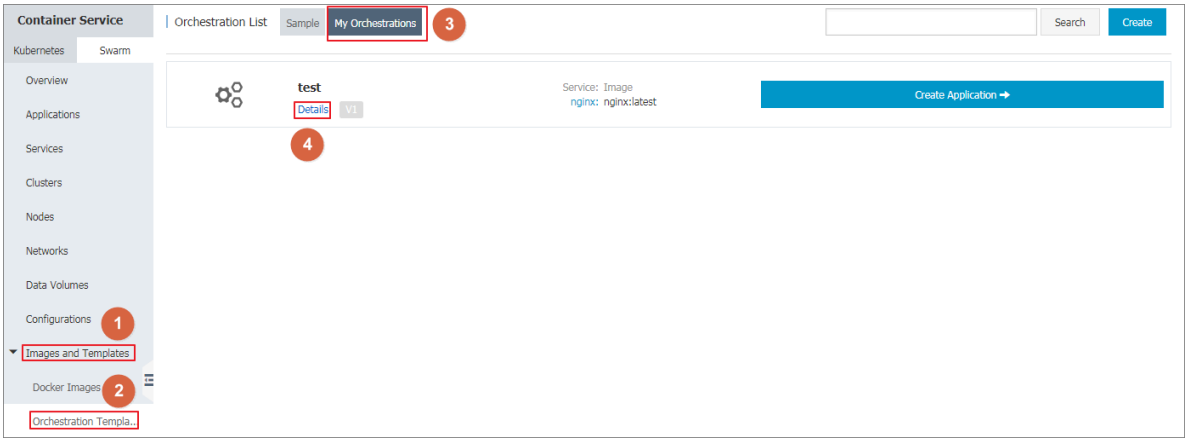
Context

You can only edit orchestration templates displayed under My Orchestrations on the Orchestration List page. To edit templates displayed under Sample, save the sample template as your own template and then edit it.

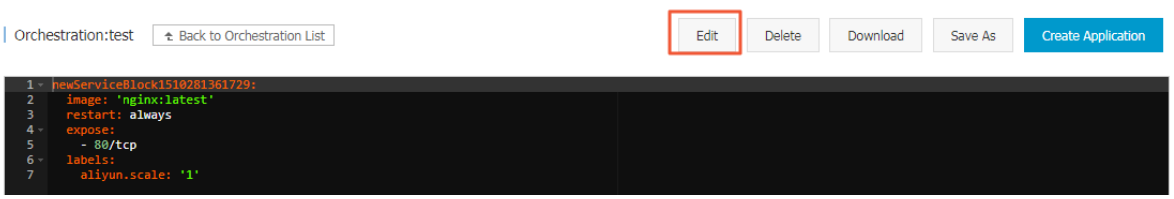
For how to save an orchestration template as a new one, see *Save an orchestration template as a new one*.

Procedure

1. Log on to the *Container Service console*.
2. n the left-side navigation pane, click Images and Templates > > Orchestration Templates.
3. Click the My Orchestrations tab and then click Details of the orchestration template you want to update.



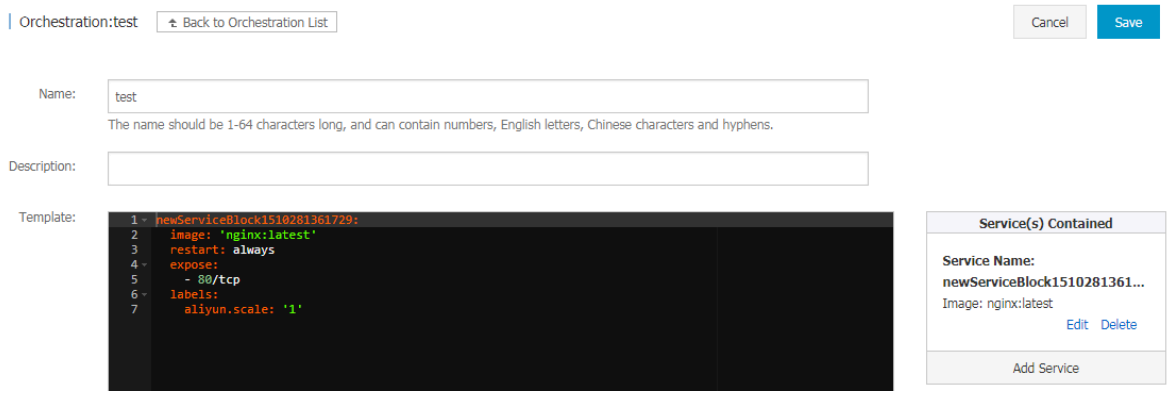
4. Click Edit in the upper-right corner.



5. Edit the template content.

   To modify a service, you can modify the content in the template directly or click Edit to modify the configurations in the appeared Create Service dialog box.

   To add another service to the orchestration template, click Add Service. The Create Service dialog box appears. Select an image and complete the other configurations. Click OK. You can modify the content in the template directly or click Delete to delete the service.



6. Click Save in the upper-right corner to save the modifications.

6 Service orchestrations

6.1 routing

The routing label configures the access domain name of a service.

Format:

```
aliyun . routing . port_ $ container_  port : [ http ://]$ domain |$
domain_pre  fix [:$ context_pa  th ]
```

Field description:

- `$ container_  port` : container port. Note: This is not the host port.
- `$ domain` : domain name. Enter a domain name.
- `$ domain_pre  fix` : domain name prefix. If you enter a domain name prefix, Container Service provides you with a test domain name and the domain name suffix is `.< cluster_id >.< region_id >. alicontain  er . com` .
- `$ context_pa  th` : requested service path. You can select services according to the requested path.

Domain name selection:

- If the HTTP protocol is used to expose the service, you can use the internal domain name (the top-level domain is `alicontain  er . com`) provided by Container Service for testing, or use your own domain name.
- If the HTTPS protocol is used, you can use only your own domain name. For example, `www . example . com` . You must modify the DNS settings to assign the domain name to the Server Load Balancer service provided by the container cluster.

Format requirements of the label statement:

- Container Service allocates a subdomain name to each cluster, and you only need to provide the domain name prefix to bind the internal domain name. The domain name prefix only indicates a domain name level and cannot be separated with periods (.).
- If you do not specify `scheme` , the HTTP protocol is used by default.

- The length of the domain name cannot exceed 128 characters. The length of the context root cannot exceed 128 characters.
- When you bind multiple domain names to the service, use semicolons (;) to separate them.
- A backend service can have multiple ports. These ports are exposed by the container. A port can only be assigned one label. Therefore, a service with multiple ports must be assigned multiple labels.

Example:

Use the routing label.

Bind the internal domain name `wordpress .< cluster_id >.< region_id >. alicontain er . com` provided by Container Service and your own domain name `http :// wp . sample . com / context` to port 80 of the Web service.

```
web :
  image : wordpress : 4 . 2
  links :
   - db : mysql
  labels :
    aliyun . routing . port_80 : wordpress ; http :// wp . sample .
com / context
db :
  image : mysql
  environmen t :
   - MYSQL_ROOT _PASSWORD = password
```

The internal domain name that you finally get is `wordpress . cd3dfe2690 56e4543acb ec5e19b01c 074 . cn - beijing . alicontain er . com` .

After starting the Web service, you can access the corresponding Web services by using the URL: `http :// wordpress . cd3dfe2690 56e4543acb ec5e19b01c 074 . cn - beijing . alicontain er . com` or `http :// wp . sample . com / context` .

To support the HTTPS service, upload the HTTPS certificate by using the Server Load Balancer console on the Alibaba Cloud website, and then bind the corresponding cluster to access the Server Load Balancer terminal.

routing.session_sticky

By using this feature, you can determine whether to maintain session sticky (session persistence) when you set the routing for a routing request. With session persistence,

during the session, each request is routed to the same backend container instead of being randomly routed to different containers.

Note:

- The setting takes effect only when you have configured `aliyun . routing . port_ $ containr_ port .`
- Simple routing session persistence is based on the Cookie mechanism. By default , the maximum expiration time of Cookie is 8 hours and the idle expiration time is 30 minutes.
- Simple routing session persistence is enabled by default.

The setting methods are as follows:

- Enable session persistence

  `aliyun . routing . session_st icky : true`

- Disable session persistence

  `aliyun . routing . session_st icky : false`

Example of a template orchestration file:

```
web :
  image : wordpress : 4 . 2
  links :
   - db : mysql
  labels :
    aliyun . routing . port_80 : wordpress ; http :// wp . sample .
com / context
    aliyun . routing . session_st icky : true
db :
  image : mysql
  environmen t :
   - MYSQL_ROOT _PASSWORD = password
```

7 Applications

7.1 Schedule an application to specified nodes

To deploy an application to specified nodes, we recommend that you use user tags and the `constraint` keyword to make the deployment configurations.

Note:

- The deployment constraint only works for newly created containers. It does not work when existing containers change the configurations.
- After you use a user tag to deploy an application, deleting the user tag does not affect the deployed application, but will affect the next deployment of the application. Proceed with caution when deleting user tags.

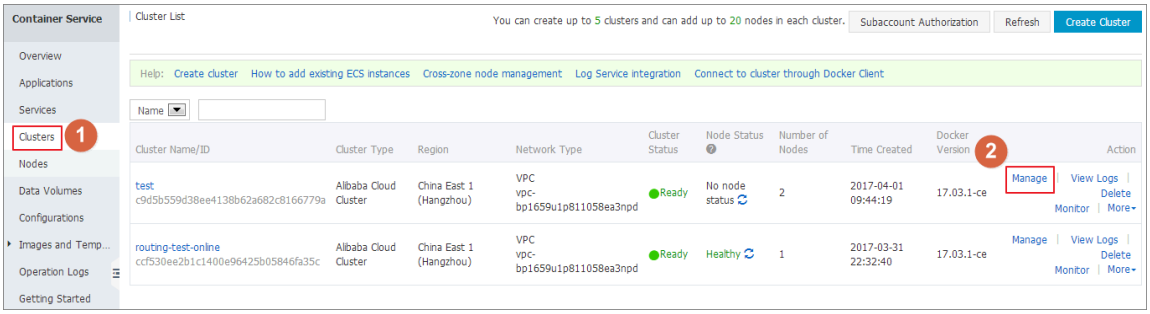
Procedure

1. Add user tags for nodes.

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

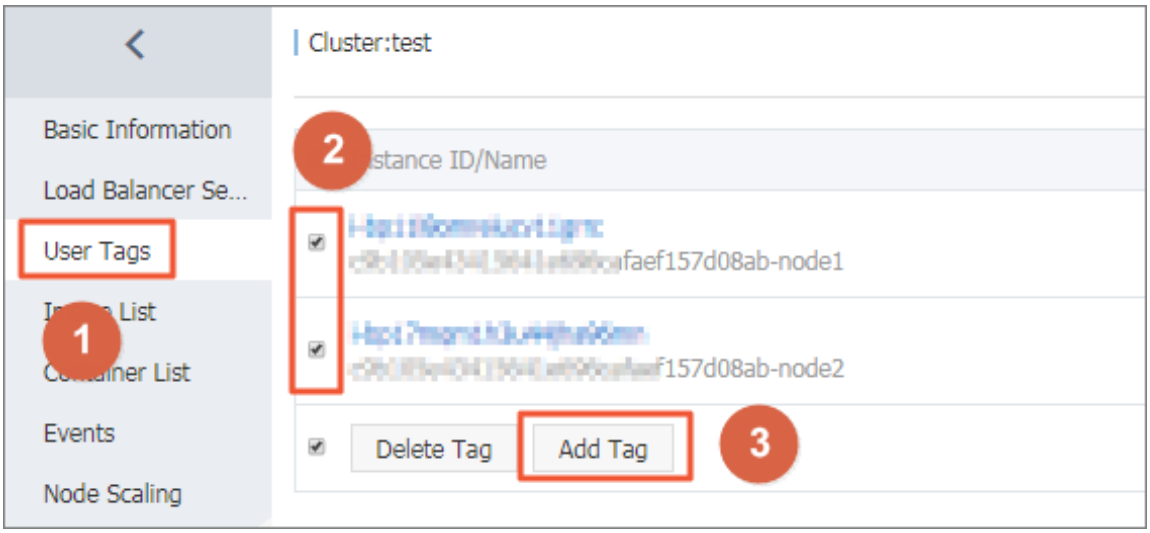
   b. Click Swarm > Clusters in the left-side navigation pane.

   c. Click Manage at the right of the cluster.

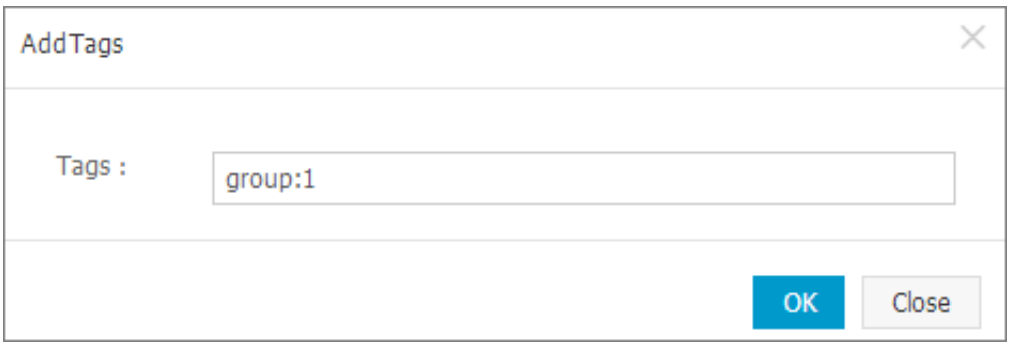


   d. Click User Tags in the left-side navigation pane.

   e. Select the nodes that you want to deploy the application and then click Add Tag.



   f. Enter your tag key and tag value, and then click OK to add user tags for the selected nodes.



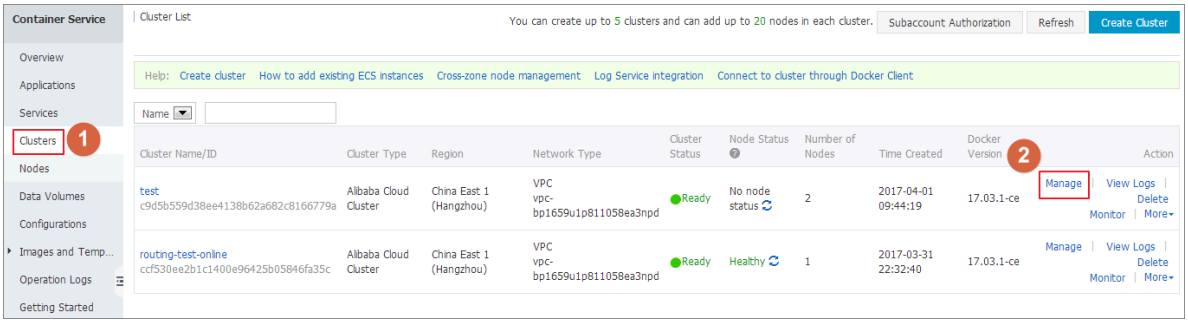
2. Create an application by clicking Create with Orchestration Template. Configure the `constraint` keyword in the template.

   For information about how to create an application, see *Create an application* .

```
environmen  t :
 -  constraint : group == 1  # Indicates  to  deploy  the
applicatio  n  on  all  the  nodes  with  the  " group : 1 "
tag
```

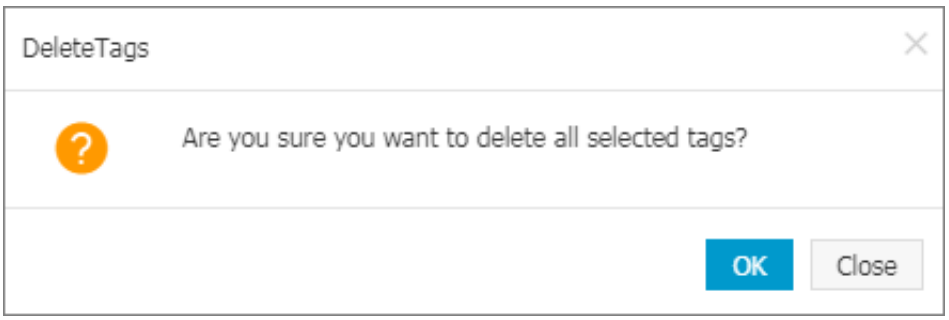
Delete a user tag

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



4. Click User Tags in the left-side navigation pane.
5. Select the nodes that you want to delete the user tags and then click Delete Tag.

6. The confirmation dialog box appears. Click OK.



8 Data volumes

9 Logs

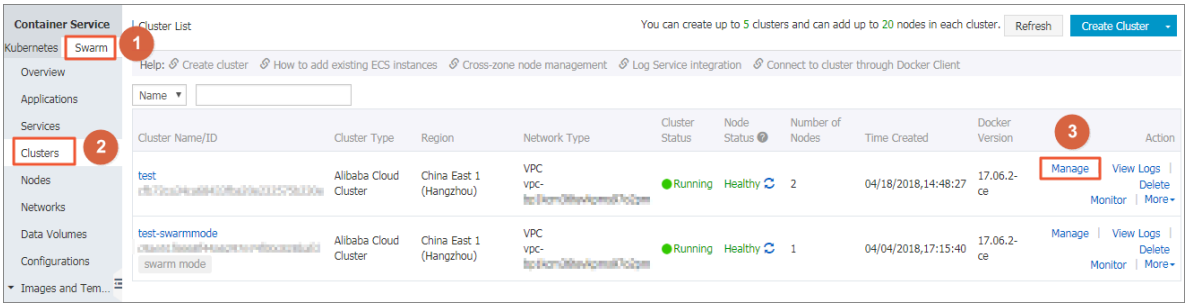
9.1 Enable Log Service

Log Service is a platform service for log scenarios. You can collect, distribute, ship, and query logs quickly without development, which is applicable to scenarios such as log transfer, monitoring, performance diagnosis, log analysis, and audit. Container Service integrates with Log Service, which allows you to send the application logs to Log Service.

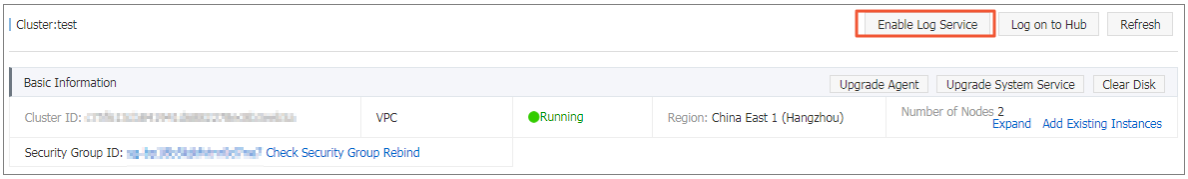
> **Note:**
> On the cluster management page, choose Enable Log Service > OK. After Log Service is successfully enabled, the log index is created for each automatically created Logstore by using the built-in Resource Access Management (RAM) account. With this feature enabled, you are charged for the Alibaba Cloud Log Service usage after configuring the following settings. For more information, see *Billing method*. Make sure you know your log volume to avoid large unexpected costs.

Enable Log Service

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

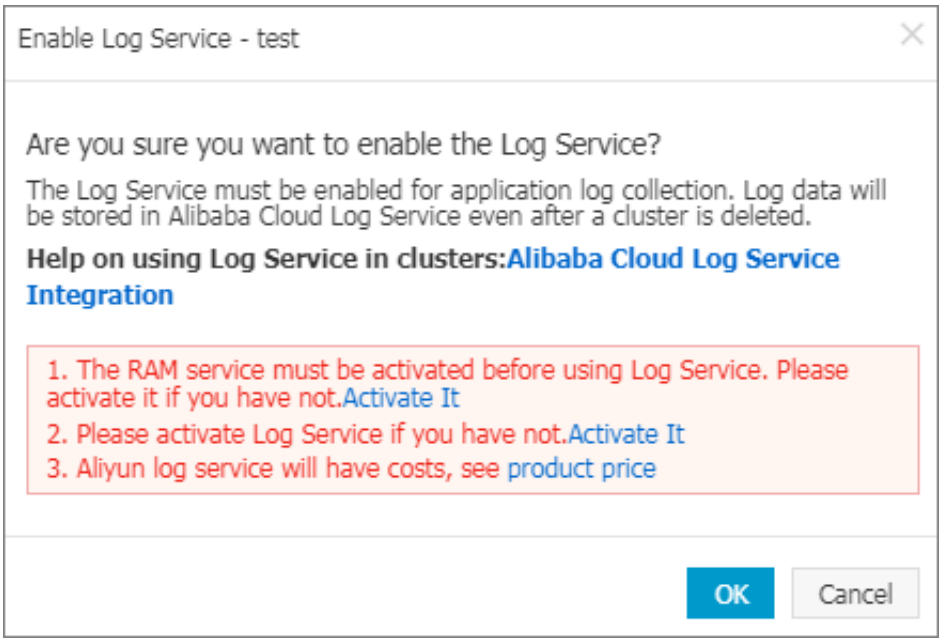


4. Click Enable Log Service in the upper-right corner.



5. In the dialog box, click OK.

   Before enabling Log Service in Container Service, activate the RAM service and Log Service first. Click Activate It to activate the RAM service and Log Service if they are not activated yet. The created Log Service project is displayed after Log Service is successfully enabled.

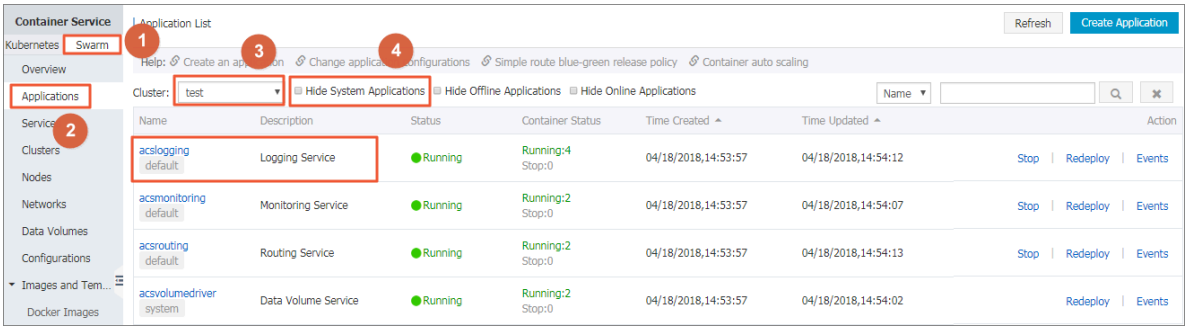


Check installation result of acslogging service

Container Service installs the Agent required by Log Service on your machine if this is the first time Log Service is enabled.You can use Log Service after the application is installed successfully. You can find this application on the Application List page. You can use Log Service after the application is installed successfully.

1. Log on to the *Container Service console*.
2. Click Applications in the left-side navigation pane.
3. Select the cluster from the Cluster list and clear the Hide System Applications check box.

   The acslogging application is successfully installed.



The system creates a corresponding project in Alibaba Cloud Log Service. You can view the project in the Log Service console. The project name contains the Container Service cluster ID.

Use Log Service in orchestration files

Most Docker applications write the logs directly to stdout, now you can do this as well (for the scenarios of writing logs to files, see *Use file logs* in the following section). After enabling Log Service, stdout logs are automatically collected and sent to Alibaba Cloud Log Service.

In the following example a WordPress application is created. It contains two services : WordPress service and MySQL service. Logs are collected to Alibaba Cloud Log Service. which contains two services: WordPress service and MySQL service. Logs are collected to Alibaba Cloud Log Service.

MySQL and WordPress

```
mysql :
  image :  mysql
  ports :
     -  80
  labels :
      aliyun . scale : " 1 "
  environmen  t :
     -  MYSQL_ROOT  _PASSWORD = password
web :
  image :  registry . aliyuncs . com / jiangjizho  ng / wordpress
  ports :
    -  80
  labels :
     aliyun . routing . port_80 :  wordpress - with - log
     aliyun . log_store_  dbstdout :  stdout   #  Collect   stdout
logs   to   the   dbstdout   Logstore .
     aliyun . log_ttl_db  stdout :  30   #  Set   the   data
retention   time   for   the   dbstdout   Logstore   to   30   days .
  links :
     -  mysql
```

In the preceding orchestration file:

- `aliyun.log_store_dbstdout:stdout` indicates to write the container standard to the Logstore `acslog-wordpress-dbstdout`. The label format is `aliyun.log_store_{name}:{logpath}`。 Wherein:
  - `name` is the name of the Alibaba Cloud Log Service Logstore. The actually created Logstore name is `acslog-${app}-${name}`.
  - `app` is the application name.
  - `logpath` is the log path in the container.
  - `stdout` is a special `logpath`, indicating the standard output.
- `aliyun.log_ttl_<logstore_name>` is used to set the data retention time (in days) for the Logstore. The value range 1–365. If left empty, logs are kept in the Logstore for two days by default.

> Note:
> The value configured here is the initial configuration value. To modify the data retention time later, modify it in the Log Service console.

You can create an application named `wordpress` in the Container Service console by using the preceding orchestration file. After the application is started, you can find the Logstore `acslog-wordpress-dbstdout` in the Log Service console, in which stores the logs of application `wordpress`.

View logs in Log Service console

After deploying an application by using the preceding orchestration file, you can view the collected logs in the Alibaba Cloud Log Service console. Log on to the Log Service console. Find the Log Service project corresponding to the cluster. You can view the Logstore `acs-wordpress-dbstdout` used in the orchestration file.

Click Search at the right of the Logstore to view the logs.

Use file logs

To write the logs directly to files (for example, `/ var / log / app . log`) instead of stdout, configure as follows:

```
aliyun . log_store_  name : / var / log / app . log
```

`name` is the Logstore name. `/ var / log / app . log` is the log path in the container.

To output multiple log files to Log Service, configure as follows to put the files under multiple directories:

```
aliyun . log_store_  s1 : / data / logs / access / access . log
aliyun . log_store_  s2 : / data / logs / error / error . log
aliyun . log_store_  s3 : / data / logs / exception /*. log  #
Wildcards   are   supported
```

Note:

Currently, multiple Logstores cannot correspond to the same log directory. The log files corresponding to the three Logstores s1, s2, and s3 in the preceding example must be under three directories.

Enable timestamp

You can select whether to add timestamp when Docker is collecting logs. Configure timestamp by using the `aliyun . log . timestamp` label in Container Service. label in Container Service. The timestamp is added by default.

- Add timestamp

  `aliyun . log . timestamp : " true "`

- Remove timestamp

  `aliyun . log . timestamp : " false "`

10 DevOps

10.1 Jenkins-based continuous delivery

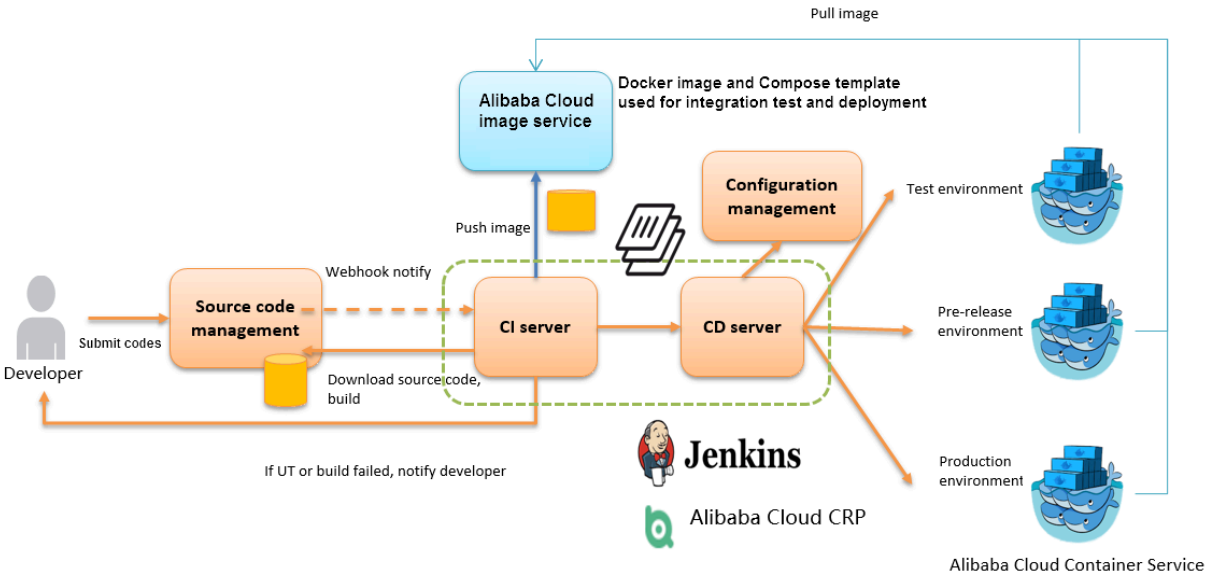
As an important step in agile development, continuous integration aims to maintain high quality while accelerating product iteration. Every time codes are updated, an automated test is performed to test the codes and function validity. The codes can only be delivered and deployed after they pass the automated test. This document mainly introduces how to integrate Jenkins, one of the most popular continuous integration tools, with Alibaba Cloud Container Service to realize automated test and image building push.

The following example demonstrates how to perform automated test and build a Docker image by using Alibaba Cloud Container Service Jenkins, which realizes high-quality continuous integration.

Background information

Every time codes are submitted to nodejs project in GitHub, Alibaba Cloud Container Service Jenkins will automatically trigger a unit test. If the test is successful, Jenkins continues to build images and then pushes them to a target image repository. Finally, Jenkins notifies you of the results by email.

A general process is as follows.



Slave-nodejs is a slave node used for unit test and building and pushing the image.

Jenkins introduction

Jenkins is an open-sourced continuous integration tool developed on Java. It monitors and triggers continuously repeated work and supports expansion of multiple platforms and plug-ins. Jenkins is an open-sourced tool featuring easy installation and interface-based management. It uses job to describe every work step, and node is a project execution environment. The master node is a default execution environment of a Jenkins job and also the installation environment for Jenkins applications.

Master/slave

Master/slave is equivalent to the server/agent concept. A master provides Web interface with which you manage the job and slave. The job can run on the master or be assigned to the slave. One master can be associated with several slaves to serve different jobs or different configurations of the same job.

Several slaves can be configured to prepare a separate test and building environment for different projects.

 Note:
The Jenkins job and project mentioned in this document all refer to a build unit of Jenkins, namely, an execution unit.

Step 1 Deploy Jenkins applications and slave nodes

The building and testing of different applications need different dependencies. The best practice is to use different slave containers with corresponding runtime dependencies and tools to perform the test and building. By using the slave images and sample templates provided by Alibaba Cloud Container Service for different environments such as Python, Node.js, and Go, you can quickly and easily generate Jenkins applications and various slave nodes, configure node information in Jenkins applications, and specify the execution nodes in the build projects so as to implement the entire continuous integration process.

 Note:
For images provided by Alibaba Cloud Container Service for developing slave nodes, see *https://github.com/AliyunContainerService/jenkins-slaves*.

1.1 Create a Jenkins orchestration template

Create a template and create the orchestration based on the following contents.

The labels supported by Alibaba Cloud Container Service Jenkins master are: 1.651.3, 2.19.2, and 2.32.2.

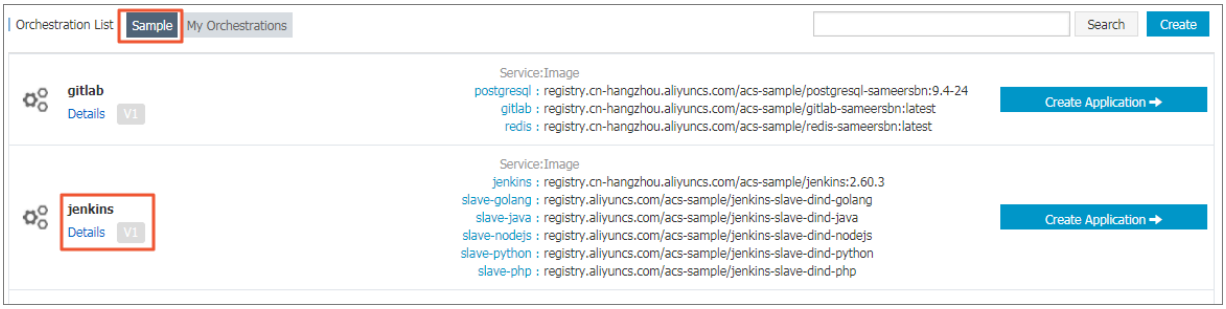
Note:
For how to create an orchestration template, see *#unique_88*.

```
jenkins :
    image : ' registry . aliyuncs . com / acs - sample / jenkins : 1
. 651 . 3 '
    volumes :
       - / var / lib / docker / jenkins :/ var / jenkins_ho  me
    restart :  always
    labels :
        aliyun . scale : ' 1 '
        aliyun . probe . url : ' tcp :// container : 8080 '
        aliyun . probe . initial_de  lay_second  s : ' 10 '
        aliyun . routing . port_8080 :  jenkins
    links :
       -  slave - nodejs
slave - nodejs :
    image : ' registry . aliyuncs . com / acs - sample / jenkins -
slave - dind - nodejs '
    volumes :
       - / var / run / docker . sock :/ var / run / docker . sock
    restart :  always
    labels :
        aliyun . scale : ' 1 '
```

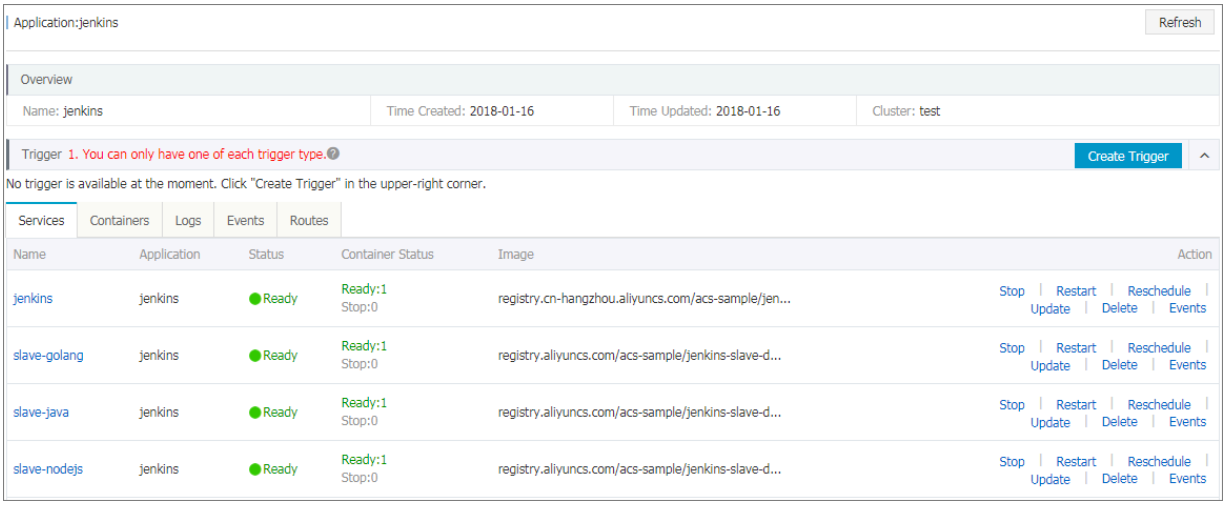
1.2 Use the template to create Jenkins application and slave node

Use the orchestration template created in the preceding section or the Jenkins sample template provided by Alibaba Cloud Container Service to create the Jenkins application and slave node.

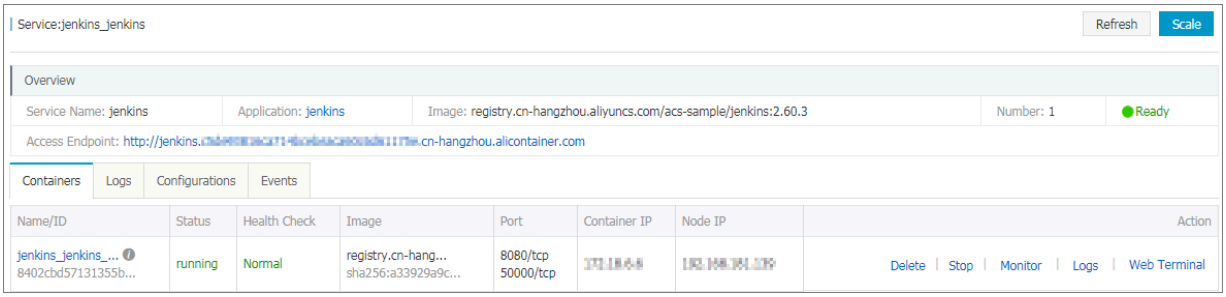
Note:
For how to create an application by using an orchestration template, see *Create an application* .



After a successful creation, the Jenkins application and slave node are displayed in the service list.



Open the access endpoint provided by Container Service to use the deployed Jenkins application.



Step 2 Realize automated test and automated build and push of image

2.1 Configure the slave container as the slave node of the Jenkins application

Open the Jenkins application. Click Manage Jenkins in the left-side navigation pane. Click Manage Nodes on the right pane. Click New Node in the left-side navigation pane. Enter the node name and then click OK. Then, complete the parameters as follows.

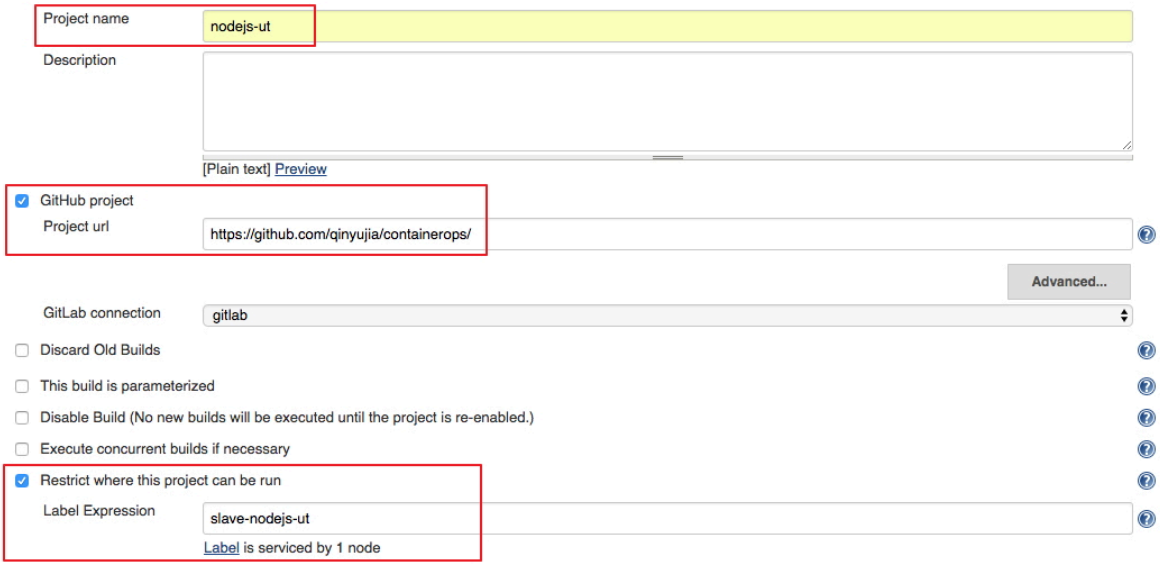
Note:

- Label is the unique identifier of the slave.
- The slave container and Jenkins container run on the Alibaba Cloud platform at the same time. Therefore, enter a container node IP address that is inaccessible to the Internet to isolate the test environment.
- When adding the credentials, use the jenkins account and password (the initial password is jenkins) in Dockerfile for the creation of the slave-nodejs image. The image Dockerfile address is *jenkins-slave-dind-nodejs*.

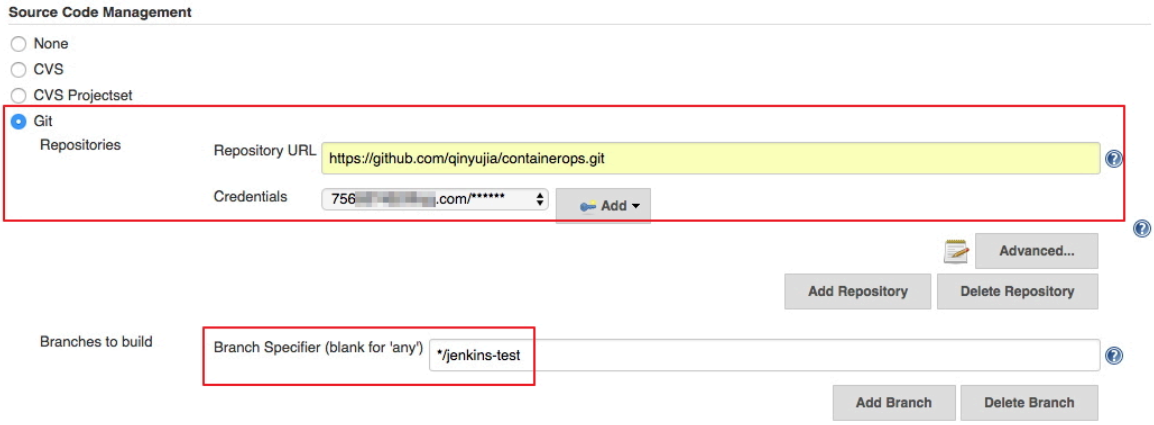
2.2 Create a project to implement automated test

1. Go back to the Jenkins home page. Click New Item in the left-side navigation pane. Enter the item name, select Freestyle project, and then click OK.

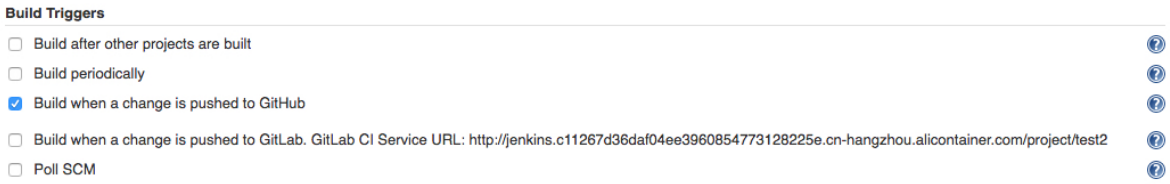
2. Enter the project name and select a node for running the project. In this example, enter the slave-nodejs-ut node prepared in the preceding section.



3. Configure the source code management and code branch. In this example, use GitHub to manage source codes.



4. Configure the build trigger. In this example, automatically trigger project execution by combining GitHub Webhooks & services.

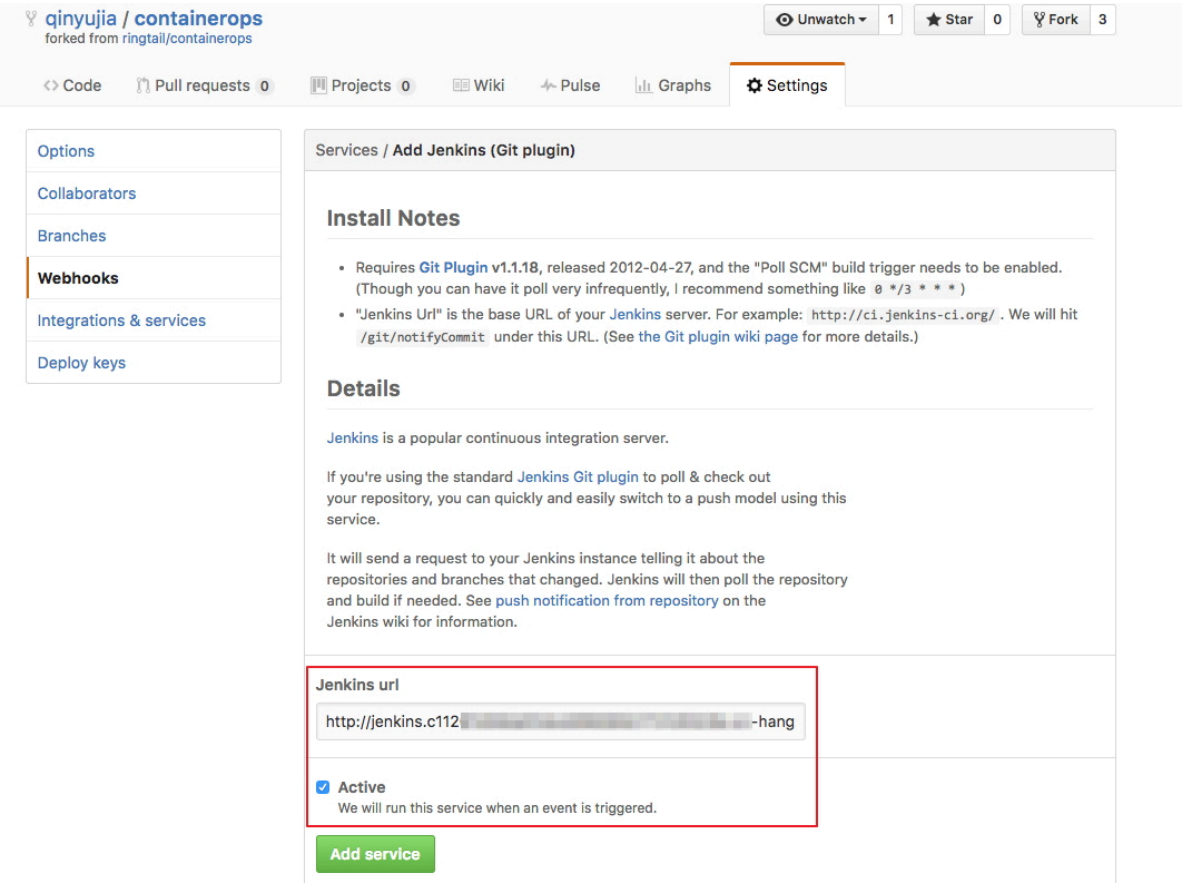


5. Add the Jenkins service hook to GitHub to implement automatic triggering.

   On the GitHub project home page, click the Settings. Click Webhooks & services, click Add Service, and then select Jenkins(Git plugin) from the drop list. In the

dialog box of `Jenkins hook url`, enter `${ Jenkins IP }/ github - webhook /`. For example:

```
http :// jenkins . cd ***************. cn - beijing . alicontain
er . com / github - webhook /
```



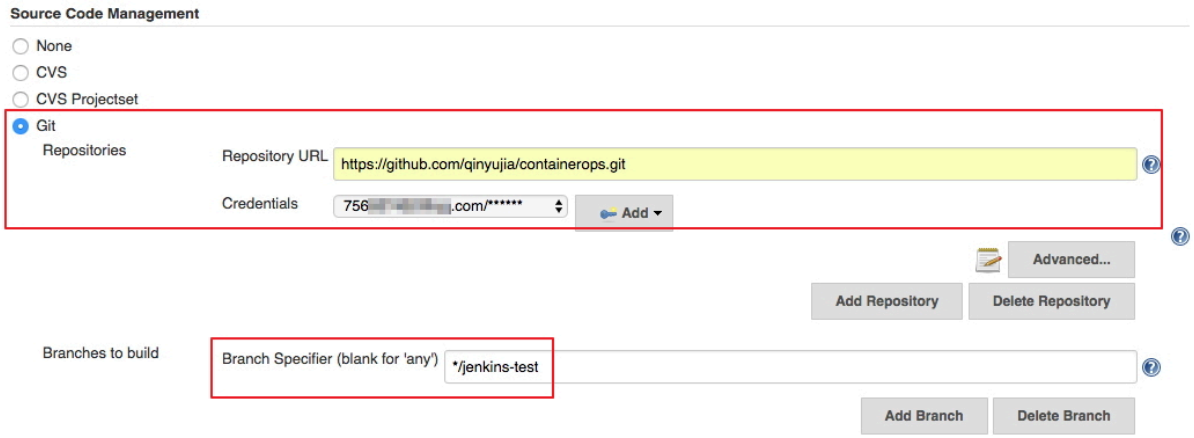
6. Add a build step of Execute shell type and write shell scripts to perform the test.

The commands in this example are as follows:

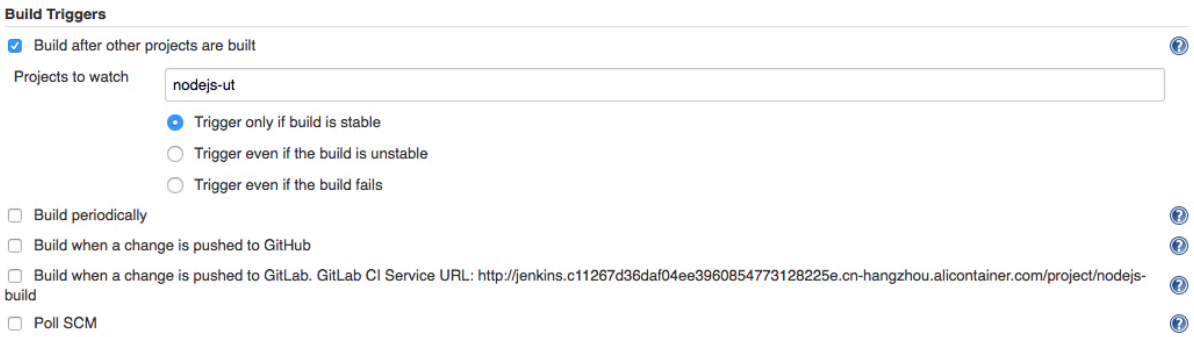
```
pwd
 ls
 cd   chapter2
 npm   test
```

SVN source code example:

Select Subversion in Source Code Management and enter the SVN repository address in the `Repository URL` field (if the Jenkins master and SVN server are in different time zones, add `@ HEAD` at the end of the repository address). Add the username and password of the SVN server in `Credentials` .



Configure the build trigger. In this example, Post-commit hook is used to automatically trigger the project execution. Enter your configured token in `Token Name` .



Log on to the SVN server. Create a *post-commit* file in the *hooks* directory of the code repository (svn-java-demo).

```
cd /home/svn/svn-java-demo/hooks
cp post-commit.tmpl post-commit
chmod 755 post-commit
```

Add the curl -u ${Jenkins_account}:${password}

```
${Jenkins_url}/job/svn/build?
        token=${token} command
```

in the <g id="1">post-commit</g> file. For example:

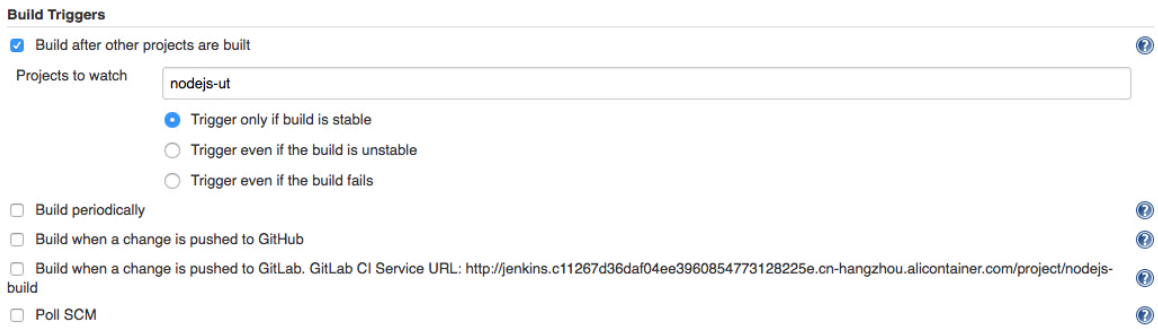
```
curl -u test:test
```

```
        http :// 127 . 0 . 0 . 1 : 8080 / jenkins / job / svn / build ?
token = qinyujia
```

.

2.3 Create a project to automatically build and push images

1. Go back to the Jenkins home page. Click New Item in the left-side navigation pane. Enter the item name, select Freestyle project, and then click OK.
2. Enter the project name and select a node for running the project. In this example, enter the slave-nodejs-ut node prepared in the preceding section.
3. Configure the source code management and code branch. In this example, use GitHub to manage source codes.
4. Add the following trigger and set to automatically build the image only after the unit test is successful.



5. Write the shell script for building and pushing images.

The commands in this example are as follows:

```
cd   chapter2
 sudo   docker   build  - t   registry . aliyuncs . com / qinyujia
- test / nodejs - demo  .
 sudo   docker   login  - u  ${ yourAccoun  t } - p  ${ yourPasswo
rd }  registry . aliyuncs . com
 sudo   docker   push   registry . aliyuncs . com / qinyujia - test
/ nodejs - demo
```

Step 3 Automatically redeploy the application

3.1 Deploy the application for the first time

Use the orchestration template to deploy the image created in step 2.3 to Container Service and create the nodejs-demo application.

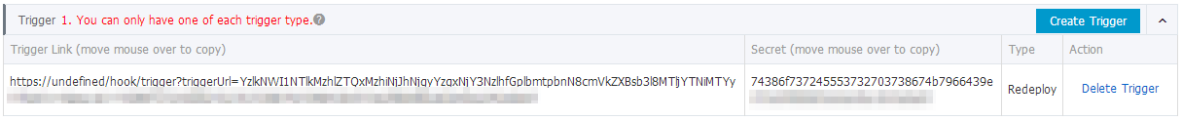
Example:

```
express :
image : ' registry . aliyuncs . com / qinyujia - test / nodejs - demo
'
expose :
   - ' 22 '
   - ' 3000 '
restart :  always
labels :
    aliyun . routing . port_3000 :  express
```

3.2 Automatic redeployment

1. Select the created application nodejs-demo and create the trigger.

> **Note:**
> For how to create a trigger, see *Triggers*.



2. Add a line to the shell script in 2.3. The address is the trigger link of the created trigger.

```
curl  ‘ https :// cs . console . aliyun . com / hook / trigger ?
triggerUrl =***==& secret =***’
```

3. Change the command in the example of 2.3 as follows:

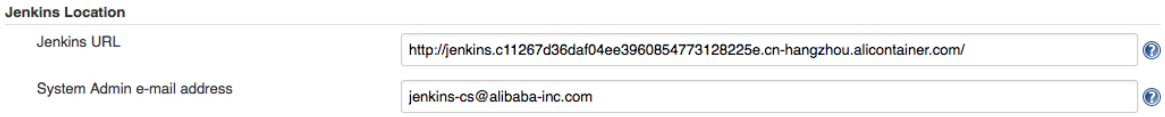
```
 cd   chapter2
 sudo   docker   build  - t   registry . aliyuncs . com / qinyujia
- test / nodejs - demo  .
 sudo   docker   login  - u  ${ yourAccoun  t } - p  ${ yourPasswo
rd }  registry . aliyuncs . com
 sudo   docker   push   registry . aliyuncs . com / qinyujia - test
/ nodejs - demo
 curl  ‘ https :// cs . console . aliyun . com / hook / trigger ?
triggerUrl =***==& secret =***’
```

After pushing the image, Jenkins automatically triggers the redeployment of the nodejs-demo application.

Step 4 Configure email notification of the results

To send the unit test or image building results to relevant developers or project execution initiators by email, perform the following configurations:

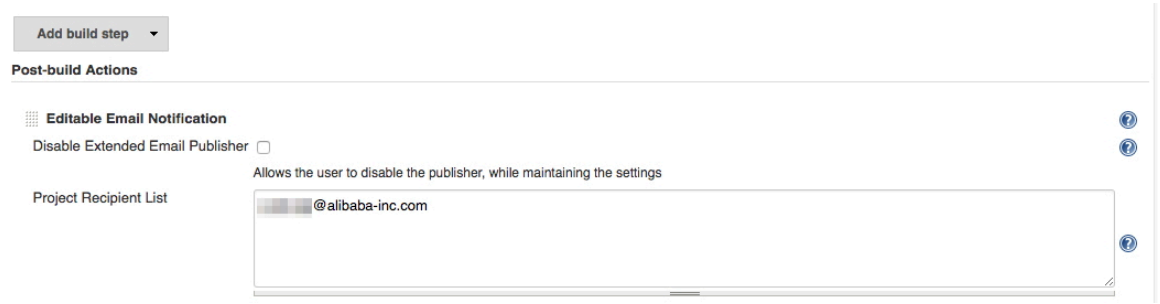
1. On the Jenkins homepage, click Manage Jenkins > Configure System, and configure the Jenkins system administrator email.



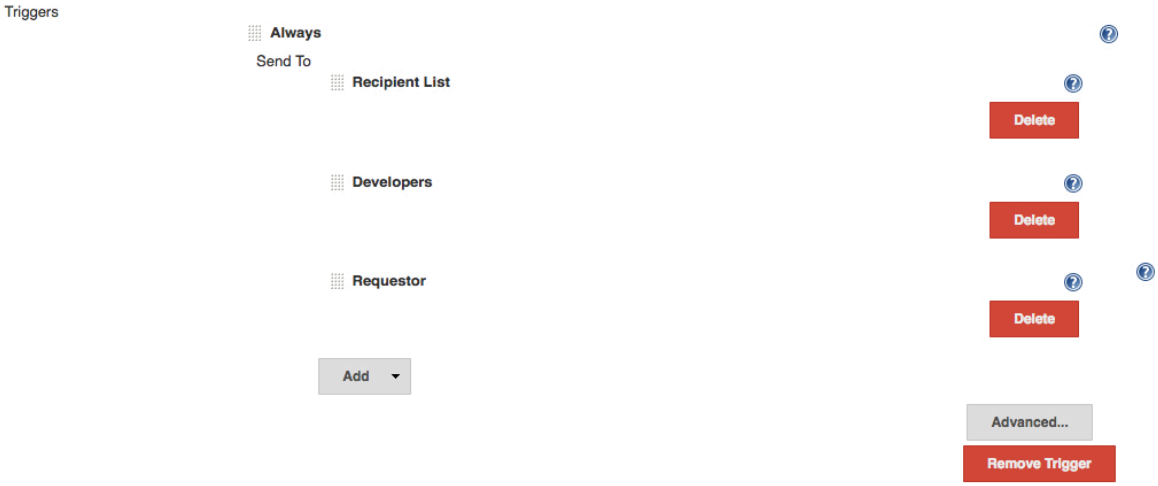
2. Install the Extended Email Notification plug-in, configure the SMTP server and other relevant information, and then set the default email recipient list, as shown in the following figure:

The preceding example shows the parameter settings of the Jenkins applicatio n system. The following example shows the relevant configurations for Jenkins projects whose results are to be pushed by email.

3. Add post-building steps in the Jenkins project, select Editable Email Notification and enter the email recipient list.



4. Add a trigger to send emails.



11 Service discovery and load balancing

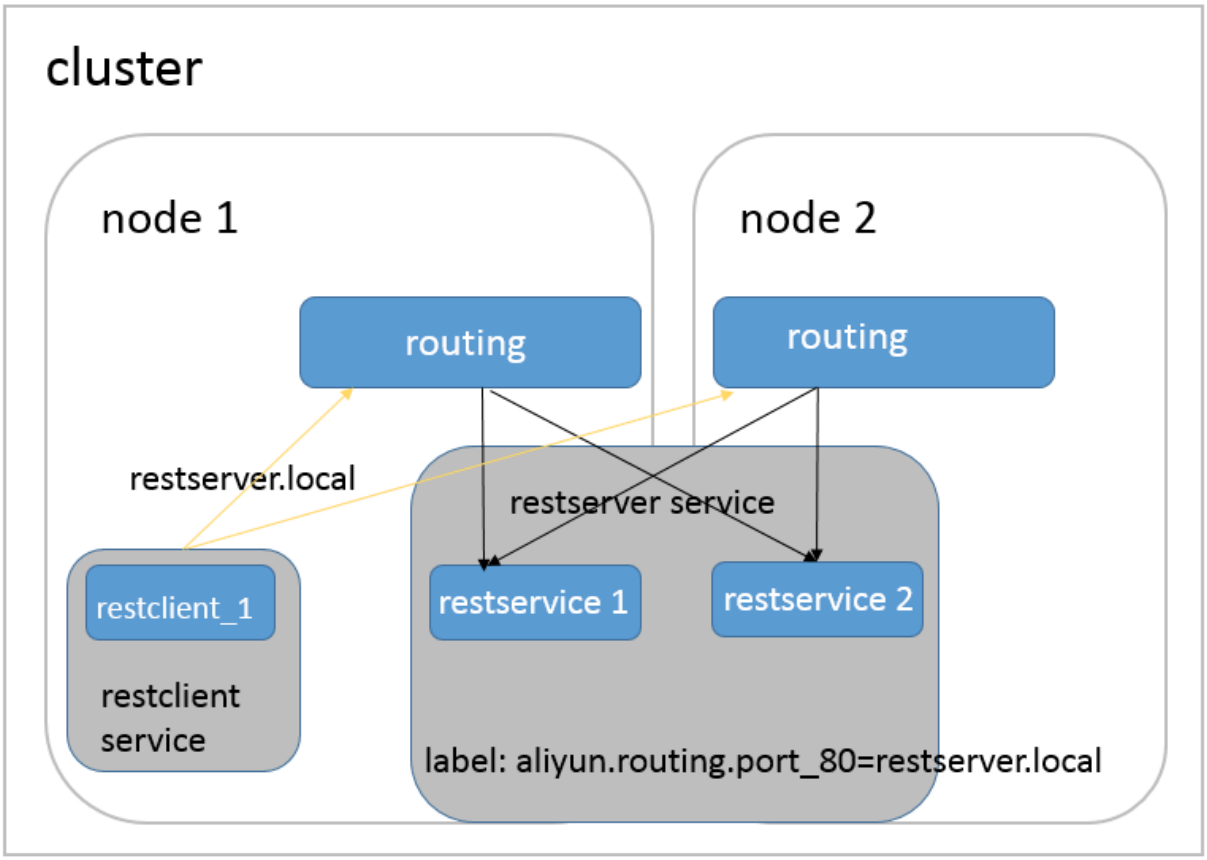
11.1 Routing and Server Load Balancer between services in a cluster

Container Service can expose the HTTP service based on domain names by using acsrouting, and work with health check to enable the automatic Server Load Balancer and service discovery. When one container malfunctions, routing will automatically remove the container that failed the health check from the backend, which achieves the automatic service discovery. However, in this way, the service is exposed to the Internet.

Then, how can automatic service discovery and Server Load Balancer be achieved between services in a cluster by using this method? The routing container of Alibaba Cloud Container Service has the function of Server Load Balancer. Use the domain name ending with `. local` to make the container can only be accessed by the other containers in the cluster, and then work with the external_links label to implement the inter-service discovery and Server Load Balancer in the cluster.

Implementation principle

1. Docker version later than 1.10 supports alias resolution in the container. In the restservice container that depends and loads on the restserver.local, the `restserver . local` domain name resolves the address of the routing container. When the restclient service initiates a request, the HTTP request is forwarded to the routing container, with `HOST` as the request header of restserver.local.
2. Routing container monitors the health status of the containers configured with `aliyun . routing . port_xxx : restserver . local` label and mounts the status to the backend of HAProxy. When HAProxy receives the HTTP request with the `restserver . local` HOST header, the request can be forwarded to the corresponding container.



Advantages

- Compared with the DNS-based method using link or hostname, the inconsistent handling of DNS cache by different clients will delay service discovery, and the DNS solution which only includes round robin cannot meet the requirements of microservice scenarios.
- Compared with other microservice discovery solutions, this solution provides a mechanism to achieve unrelated service discovery and Server Load Balancer, which can be used without any modification on the server side or client application .
- In decoupling service lifecycle, every microservice can adopt a Docker Compose template for independent deployment and update. Only a virtual domain name is required to achieve dynamic mutual binding.

Orchestration example

In the following orchestration example, add the `aliyun . routing . port_80 : restserver . local` label to the restserver service to make sure only the containers in the cluster can access this domain name. Then, configure `external_l inks` for the restclient service, pointing to the restserver.local

domain name. The restclient service can use this domain name to access the restserver service, and work with health check to implement automatic service discovery.

```
restserver : #  Simulate   the   rest   service .
  image :  nginx
  labels :
    aliyun . routing . port_80 :  restserver . local  #  Use   the
  local   domain   name   and   only   the   containers   in   the
cluster   can   access   this   domain   name .
    aliyun . scale : " 2 " #  Expand   two   instances   to
simulate   the   Server   Load   Balancer .
     aliyun . probe . url : " http :// container : 80 " #  Define
the   container   health   check   policy   as   http   and   the
port   as   80 .
    aliyun . probe . initial_de  lay_second  s : " 2 " #  The
health   check   starts   two   seconds   after   the   container
is   started .
     aliyun . probe . timeout_se  conds : " 2 " #  The   timeout
  for   health   check .  A   container   is   considered   as
unhealthy   if   no   result   is   returned   in   two   seconds .
restclient : #  Simulate   the   rest   service   consumer .
  image :  registry . aliyuncs . com / acs - sample / alpine : 3 . 3
  command : " sh  - c  ' apk   update ;  apk   add   curl ;  while
true ;  do   curl  -- head   restserver . local ;  sleep   1 ;  done
'" #  Access   the   rest   service   and   test   the   Server
Load   Balancer .

  tty :  true
  external_l  inks :
    - " restserver . local " #  Specify   the   link   service
domain   name .   Make   sure   that   you   set   external_l  inks
.  Otherwise ,  the   access   fails .
```

The following restclient service logs show that the HTTP request of restclient curl is routed to the containers of different rest services. The container ID is `053cb232fd fbcb5405ff 791650a074 6ab77f26cc e74fea2320 075c2af55c 975f` and `b8c36abca5 25ac7fb02d 2a9fcaba8d 36641447a7 74ea956cd9 3068419f17 ee3f`.

```
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
49 . 066803626Z   Server :  nginx / 1 . 11 . 1
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43
: 49 . 066814507Z   Date :  Fri ,  01   Jul   2016   06 : 43 : 49
GMT
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
49 . 066821392Z   Content - Type :  text / html
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
49 . 066829291Z   Content - Length :  612
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
49 . 066835259Z   Last - Modified :  Tue ,  31   May   2016   14 : 40
: 22   GMT
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
49 . 066841201Z   ETag : " 574da256 - 264 "
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
49 . 066847245Z   Accept - Ranges :  bytes
```

```
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 :
43 : 49 . 066853137Z   Set - Cookie :  CONTAINERI  D = 053cb232fd
fbcb5405ff  791650a074  6ab77f26cc  e74fea2320  075c2af55c  975f ;
path =/
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
50 . 080502413Z   HTTP / 1 . 1   200   OK
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
50 . 082548154Z   Server :  nginx / 1 . 11 . 1
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43
: 50 . 082559109Z   Date :  Fri ,  01   Jul   2016   06 : 43 : 50
GMT
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
50 . 082589299Z   Content - Type :  text / html
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
50 . 082596541Z   Content - Length :  612
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
50 . 082602580Z   Last - Modified :  Tue ,  31   May   2016   14 : 40
: 22   GMT
internal - loadbalanc  e_restclie  nt_1    2016 - 07 - 01T06 : 43 :
50 . 082608807Z   ETag : " 574da256 - 264 "
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 : 43 :
50 . 082614780Z   Accept - Ranges :  bytes
internal - loadbalanc  e_restclie  nt_1  |  2016 - 07 - 01T06 :
43 : 50 . 082621152Z   Set - Cookie :  CONTAINERI  D = b8c36abca5
25ac7fb02d  2a9fcaba8d  36641447a7  74ea956cd9  3068419f17  ee3f ;
path =/
```