ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

Container Service for Kubernetes

User Guide

Product Version: v3.16.2

Document Version: 20220916

Alibaba Cloud

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 because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should 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 this document based on the "status quo", "being defective", and "existing functions" of its products and services. 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 take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, 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 contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, 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 this document 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 directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice 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.
Warning	A warning notice 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 restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Announcements

1.1. Container Service support for Kubernetes 1.20

Container Service strictly conforms to the terms of the Certified Kubernetes Conformance Program. This topic lists the changes that Container Service has made to support Kubernetes 1.20.

Version updates

All Container Service components have been upgraded and optimized to support Kubernetes 1.20.

Core component	Version	Description
Kubernetes	1.20.11	• Before you upgrade a Container Service cluster to Kubernetes 1.20 or later, make sure that the required subject alternative names (SANs) are included in the self-signed server certificates of the admission webhooks in the cluster. For more information, see the sample Helm chart. • The selfLink field is deprecated. For more information, see Stop setting SelfLink in kube-apiserver. • By default, the `node-role.kubernetes.io/control-plane` label is added by Container Service to the master nodes of a dedicated Kubernetes cluster. The `node-role.kubernetes.io/master` label is deprecated in Kubernetes versions later than 1.20.
Docker Runtime	19.03.15	None
etcd	3.4.3	None
CoreDNS	1.7.0	The names of metrics are updated. If your monitoring system is reliant on CoreDNS metrics, you must update the metric names. For more information, see Metric changes.

Version details

Resource changes and deprecation

- By default, the `node-role.kubernetes.io/control-plane` label is added by Container Service to the master nodes of a dedicated Kubernetes cluster. The `node-role.kubernetes.io/master` label is deprecated in Kubernetes versions later than 1.20.
- The selfLink field is deprecated. For more information, see Stop setting SelfLink in kube-apiserver.
- The `extensions/v1beta1` and `networking.k8s.io/v1beta1` API versions are no longer used to manage Ingresses and IngressClasses, and will be deprecated in Kubernetes versions later than 1.22. Use `networking.k8s.io/v1` instead.

> **Note** By default, the NGINX Ingress controller is installed in Container Service clusters. This component enables you to use the networking.k8s.io/v1beta1 API version to manage Ingresses and IngressClasses.

- The required SANs must be included in the self-signed server certificates of the admission webhooks in Container Service clusters. Before you upgrade a Container Service cluster to Kubernetes 1.20 or later, make sure that the required SANs are included in the self-signed server certificates of the admission webhooks in the cluster. For more information, see the sample Helm chart.

Feature upgrades

- The issue that exec probes do not time out based on the timeout settings is fixed for kubelet. The default timeout period for exec probes is now 1 second, which may be short for some exec probes. If the timeout period is not specified for exec probes, we recommend that you specify the default timeout period.
- The API Priority and Fairness feature (APF) is a feature of Kubernetes in public preview and is enabled by default. You can use this feature to limit and prioritize requests. For more information, see API Priority and Fairness.
- By default, the EndpointSlice feature is enabled. In Kubernetes 1.19 and later, the EndpointSlice feature is automatically enabled by kube-proxy to support large-scale clusters. For more information, see EndpointSlices.
- Immutable ConfigMaps and Secrets are supported. The immutable ConfigMaps and Secrets feature is in public preview. If a ConfigMap or Secret is set to immutable, it cannot be modified. This reduces the load on kube-apiserver. For more information, see Immutable ConfigMaps.

Enhancements to Kubernetes 1.20

Control plane improvements

- Observability. Metrics are collected to monitor request operations and watch operations. This improves the observability of control plane components.
- Stability. Protection is provided to defend etcd against excessive requests when a cluster is started. This improves system stability.
- Performance optimizations. Indexes are added to accelerate the processing of list requests. This reduces the CPU usage of kube-apiserver.

References

- CHANGELOG-1.20.md
- CHANGELOG-1.19.md

1.2. Container Service support for Kubernetes 1.18

Container Service strictly conforms to the terms of the Certified Kubernetes Conformance Program. This topic describes the changes that Container Service has made to support Kubernetes 1.18.

Version upgrades

Container Servcie has upgraded and optimized all of its components to support Kubernetes 1.18.8.

Key component	Version	Description
Kubernetes	1.18.8	Some frequently used API versions are deprecated in Kubernetes 1.18. Before you upgrade a Kubernetes cluster, we recommend that you upgrade the deprecated API versions that are listed in this topic.
Docker	19.03.5 (containerd 1.2.10)	No.
etcd	3.4.3	No.
CoreDNS	1.6.7	No.

Version details

- **Resource changes and deprecation**

  The following APIs are deprecated in Kubernetes 1.18:

  - The APIs apps/v1beta1 and apps/v1beta2 of all the resources are replaced by **apps/v1**.
  - The API extensions/v1beta1 of DaemonSets, Deployments, and ReplicaSets is replaced by **apps/v1**.
  - The API extensions/v1beta1 of Networkpolicies resources is replaced by **networking.k8s.io/v1**.
  - The API extensions/v1beta1 of pod security policies is replaced by **policy/v1beta1**.

  The label that specifies the regions of a node is changed to topology.kubernetes.io/region. The label that specifies the zone of a node is changed to topology.kubernetes.io/zone. We recommend that you update the related configurations for your workloads.

- **Feature upgrades**

  - Server-side Apply Beta 2 is introduced. You can view the relationships between the configuration items of a resource in the metadata.managedFields field of the resource.
  - The Node Local DNS Cache feature is released to improve the DNS availability and performance of your cluster.
  - The Volume Snapshot feature is in public preview and supports operations such as data volume backup, recovery, and scheduled backup.

Container Service upgrades for Kubernetes 1.18.8

In Kubernetes 1.18.8, Container Service enables the following feature in the kubelet configuration file: Users who use raw data volumes can upgrade clusters without the need to drain the nodes.

1.3. Vulnerability fixed: CVE-2021-1056 in NVIDIA GPU drivers

NVIDIA has reported the CVE-2021-1056 vulnerability, which is related to device isolation and NVIDIA GPU drivers. Elastic GPU Service nodes that are deployed in a Container Service cluster may also be exposed to this vulnerability. This topic describes the background information, affected versions, and fixes of this vulnerability.

Context

The CVE-2021-1056 vulnerability is related to device isolation and NVIDIA GPU drivers. This vulnerability allows an attacker to gain access to all GPU devices on a node by creating character device files in non-privileged containers that run on this node.

For more information about this vulnerability, see CVE-2021-1056.

Affected versions

The affected versions of NVIDIA GPU drivers are listed in the following figure based on the information published on the NVIDIA official website. For more information, see NVIDIA official website.

CVE IDs Addressed	Software Product	Operating System	Driver Branch	Affected Versions	Updated Driver Version
CVE-2021-1052 CVE-2021-1053	GeForce	Linux	R460	All versions prior to 460.32.03	460.32.03
			R450	All versions prior to 450.102.04	450.102.04
	NVIDIA RTX/Quadro, NVS	Linux	R460	All versions prior to 460.32.03	460.32.03
			R450	All versions prior to 450.102.04	450.102.04
	Tesla	Linux	R460	All versions prior to 460.32.03	460.32.03
			R450	All versions prior to 450.102.04	450.102.04
CVE-2021-1056	GeForce	Linux	R460	All versions prior to 460.32.03	460.32.03
			R450	All versions prior to 450.102.04	450.102.04
	NVIDIA RTX/Quadro, NVS	Linux	R460	All versions prior to 460.32.03	460.32.03
			R450	All versions prior to 450.102.04	450.102.04
			R390	All version prior to 390.141	390.141
	Tesla	Linux	R460	All versions prior to 460.32.03	460.32.03
			R450	All versions prior to 450.102.04	450.102.04
			R418	All versions prior to 418.181.07	418.181.07

- If you selected a custom NVIDIA driver or updated an NVIDIA driver, check whether the NVIDIA driver that you installed is affected by this vulnerability based on the preceding figure.
- If you use the NVIDIA driver that is automatically installed by Container Service, you must check whether the Kubernetes version of your cluster is affected by this vulnerability. The following Kubernetes versions are affected by this vulnerability:
  - ACK 1.16.9-aliyun.1. By default, the NVIDIA driver of version 418.87.01 is installed in clusters of this Kubernetes version.
  - ACK 1.18.8-aliyun.1. By default, the NVIDIA driver of version 418.87.01 is installed in clusters of this Kubernetes version.

> **Note** The NVIDIA GPU drivers that are installed by default in clusters of other Kubernetes versions are not affected. The Container Service team of Alibaba Cloud will keep you informed of further CVE content updates and help you fix the vulnerability.

Check the version of the NVIDIA driver on a GPU-accelerated node

Log on to the GPU-accelerated node and run the following command to query the version of the NVIDIA driver.

> **Note** For more information about how to log on to a GPU-accelerated node, see the *Use VNC to connect to and log on to an instance* chapter of the *Elastic Compute Service (ECS) User Guide*.

```
nvidia-smi
```

Expected output:

```
Fri Apr 16 10:58:19 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:07.0 Off |                    0 |
| N/A   34C    P0    37W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

The output indicates that the version of the NVIDIA driver is 418.87.01.

Fixes

Notice When you upgrade the NVIDIA driver for a node, the node must be restarted. This interrupts services that are deployed on the node.

Upgrade the NVIDIA driver based on the preceding figure.

Note For more information about how to upgrade the NVIDIA driver, see Upgrade the NVIDIA driver on a GPU node.

- If your NVIDIA driver belongs to the R390 branch, upgrade it to version 390.141.
- If your NVIDIA driver belongs to the R418 branch, upgrade it to version 418.181.07.
- If your NVIDIA driver belongs to the R450 branch, upgrade it to version 450.102.04.
- If your NVIDIA driver belongs to the R460 branch, upgrade it to version 460.32.03.

2.What is Container Service?

Container Service provides high-performance, scalable, and enterprise-class management service for Kubernetes containerized applications throughout the application lifecycle.

Container Service simplifies the deployment and scaling operations on Kubernetes clusters. Integrated with services such as virtualization, storage, network, and security, Container Service aims to provide the optimal cloud environment for Kubernetes containerized applications. Alibaba Cloud is a Kubernetes Certified Service Provider (KCSP). As one of the first services to participate in the Certified Kubernetes Conformance Program, Container Service provides you with professional support and services.

3.ACK@Edge overview

ACK@Edge is released for commercial use. ACK@Edge is a cloud-managed solution that is provided by Container Service to coordinate cloud and edge computing. This topic describes the background and features of edge Kubernetes clusters.

Overview

With the rapid growth of smart devices connected to the Internet and the advent of 5G and IoT, computing and storage services provided by traditional cloud computing platforms can no longer satisfy the needs of edge devices for time-efficient computing, larger storage capacity, and enhanced computing capacity. Edge Kubernetes clusters are intended for bringing cloud computing to edges (clients). Edge Kubernetes clusters can be created, managed, and maintained in the Container Service console. This is the trend of cloud computing.

An edge Kubernetes cluster is a standard, secure, and highly-available Kubernetes cluster deployed in the cloud. This type of cluster is integrated with features of Alibaba Cloud, such as virtualization, storage, networking, and security. This simplifies the management and maintenance of clusters and allows you to focus on your business development. ACK@Edge provides the following features:

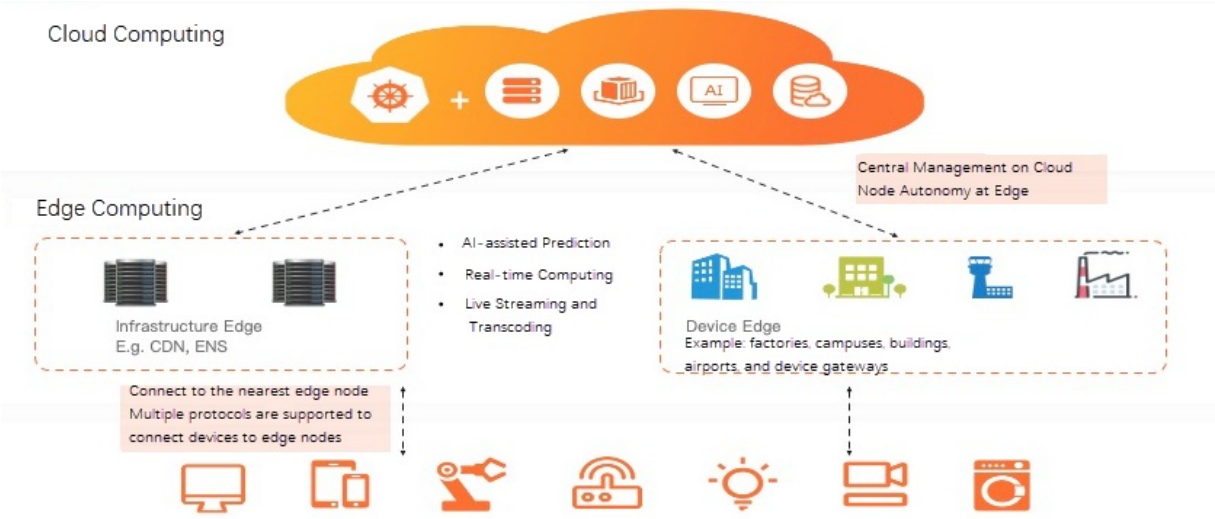
- Allows you to build a cloud-native infrastructure for edge computing with a few clicks.
- Allows you to quickly connect edge computing resources to the cloud. These resources include IoT gateway devices, terminals, Content Delivery Network (CDN) resources, and data centers.
- Applies to diverse scenarios, such as edge intelligence, intelligent buildings, intelligent factories, audio and video live streaming, online education, and CDN.

Edge Kubernetes clusters support features such as node autonomy, cell-based management, and native APIs for the management and maintenance of resources at the edge side. To use these features, you do not need to rewrite the logic of your services. This provides a native and centralized method for application lifecycle management and resource scheduling in edge computing scenarios.

Features

Edge Kubernetes clusters provide the following features to support lifecycle management for containerized applications and resources in edge computing scenarios:

- Allows you to create highly available edge Kubernetes clusters with a few clicks and provides lifecycle management on edge Kubernetes clusters, such as scaling cloud nodes, adding edge nodes to clusters, upgrading, logging, and monitoring. You can perform the preceding operations in the Container Service console.
- Supports access to various heterogeneous resources, such as data centers and IoT devices. Hybrid scheduling of heterogeneous resources is also supported.
- Supports node autonomy and network autonomy to ensure the reliability of edge nodes and services in edge computing scenarios where the network connection is weak.
- Supports reverse tunneling for management and maintenance of edge nodes.


Cloud Computing
AI
Central Management on Cloud
Node Autonomy at Edge
Edge Computing
Infrastructure Edge
E.g. CDN, ENS
AI-assisted Prediction
Real-time Computing
Live Streaming and Transcoding
Device Edge
Example: factories, campuses, buildings, airports, and device gateways
Connect to the nearest edge node
Multiple protocols are supported to connect devices to edge nodes

4.Planning and preparation

Before you start using Container Service, you need to create cloud resources such as VPC networks, VSwitches, disks, and OSS buckets based on your application requirements.

Before you create a Kubernetes cluster, make the following preparations:

- **Create a VPC network** (optional)

  To create a cluster in an existing VPC network, you must create the VPC network and VSwitches in advance.

- **Create a volume** (optional)

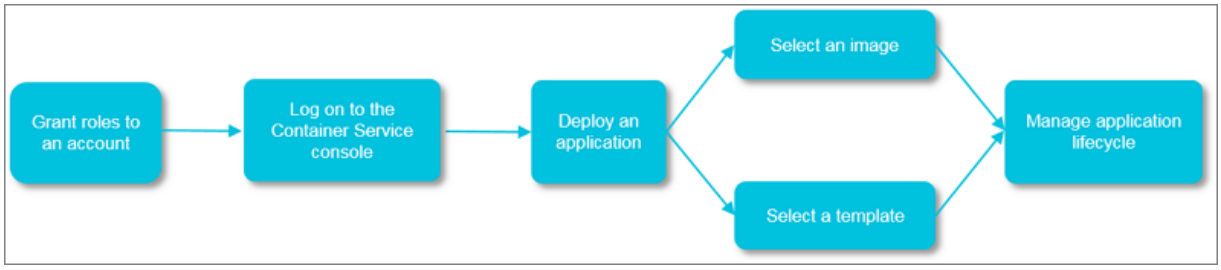
  To create a stateful application with network storage, you must create disks or OSS buckets in advance.

5.Quick start

5.1. Procedure

You can perform the following steps to use the Container Service service.

The following diagram shows the procedure to use the Container Service service.



Step 1: Authorize the default role

Authorize the default role of Container Service to perform operations on the resources that belong to the specified organization.

Step 2: Log on to the Container Service console

Log on to the Container Service console. For more information, see Log on to the Container Service console.

Step 3: Create an Container Service cluster

Set the network environment and the number of nodes, and configure node details.

Step 4: Deploy an application by using an image or orchestration template

You can use an existing image or orchestration template, or create a new image or orchestration template.

To create an application that consists of services based on different images, use an orchestration template.

Step 5: Manage the application lifecycle

5.2. Log on to the Container Service console

You can perform the following steps to log on to the Container Service console.

Prerequisites

- Before you log on to the Apsara Uni-manager Management Console, the endpoint of the console is obtained from the deployment staff.
- We recommend that you use Google Chrome.

Procedure

1. Enter the URL of the Apsara Uni-manager Management Console in the address bar and press the Enter key.

2. Enter your username and password.

   Obtain the username and password that you use to log on to the Apsara Uni-manager Management Console from the operations administrator.

   > **Note** The first time that you log on to the Apsara Uni-manager Management Console, you must change the password of your account. For security purposes, your password must meet the minimum complexity requirements. The password must be 10 to 32 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters
   > - Digits
   > - Special characters, including exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%)

3. Click **Log On**.
4. If your account has multi-factor authentication (MFA) enabled, perform corresponding operations in the following scenarios:
   - You log on to the Apsara Uni-manager Management Console for the first time after MFA is enabled by the administrator:
     a. On the Bind Virtual MFA Device page, bind an MFA device.
     b. Enter the username and password again as in Step 2 and click **Log On**.
     c. Enter a six-digit MFA verification code and click **Authenticate**.
   - You have enabled MFA and bound an MFA device:

     Enter a six-digit MFA verification code and click **Authenticate**.

     > **Note** For more information, see the *Bind a virtual MFA device to enable MFA* topic in *Apsara Uni-manager Management Console User Guide*.

5. In the top navigation bar, choose **Products > Elastic Computing > Container Service for Kubernetes**.
6. On the Container Service page, select **Access with Authorized Role** or **Access as Administrator**.
   - **Access with Authorized Role**: The system accesses the Container Service console by using an authorized account.
   - **Access as Administrator**: The system accesses the Container Service console as the organization administrator.

5.3. Create a Kubernetes cluster

This topic describes how to create a Kubernetes cluster in the Container Service console.

Procedure

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

3. On the **Dedicated Kubernetes** tab of the **Create Cluster** page, set the following parameters.

Parameter	Description
Cluster Name	Enter a name for the cluster. The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-). **Note** The cluster name must be unique among clusters that belong to the same Alibaba Cloud account.
Resource Set	Select a resource set. A resource set is a container used to store resources. Each resource must belong to a resource set. After you select a resource set, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource set.
Region	Select the region where you want to deploy the cluster.
VPC	You can select a VPC from the drop-down list. ◦ If the specified VPC has a NAT gateway, Container Service uses this NAT gateway. ◦ If the VPC does not have a NAT gateway, the system automatically creates one. If you do not want the system to create a NAT gateway, clear **Configure SNAT for VPC**. **Note** If you disallow the system to automatically create a NAT gateway and want the VPC to access the Internet, you must manually associate the VPC with a NAT gateway or create SNAT rules for the VPC.
vSwitch	Select vSwitches. You can select up to three vSwitches that are deployed in different **zones**.
Kubernetes Version	Select a Kubernetes version.
Container Runtime	You can select Docker or Sandboxed-Container.

Parameter	Description
Master Configurations	Set the Instance Type and System Disk parameters: ○ Master Node Quantity: You can add up to three master nodes. ○ Instance Type: You can select multiple instance types. For more information, see *Instance families and instance types* in the *ECS* documentation. ○ System Disk: **SSD Disk**, **Ultra Disk**, and **Enhanced SSD** are supported. **Note** You can select **Enable Backup** to back up disk data.
Worker Instance	You can select **Create Instance** or **Add Existing Instance**.
Worker Configurations	If **Worker Instance** is set to **Create Instance**, set the following parameters: ○ Instance Type: You can select multiple instance types. For more information, see *Instance families and instance types* in the *ECS* documentation. ○ Selected Types: The selected instance types are displayed. ○ Quantity: Set the number of worker nodes. ○ System Disk: **SSD Disk**, **Ultra Disk**, and **Enhanced SSD** are supported. **Note** You can select **Enable Backup** to back up disk data. ○ Mount Data Disk: **SSD Disk**, **Ultra Disk**, and **Enhanced SSD** are supported. **Note** ■ You can select **Encrypt Disk** to encrypt disks. ■ You can select **Enable Backup** to back up disk data.
Operating System	The CentOS and Alibaba Cloud Linux operating systems are supported.

Parameter	Description
Password	Set a password that is used to log on to the nodes. **Note** The password must be 8 to 30 characters in length, and must contain at least three of the following types of character: uppercase letters, lowercase letters, digits, and special characters.
Confirm Password	Enter the password again.
Network Plug-in	Flannel and Terway are supported. By default, Flannel is selected.
Pod CIDR Block and Service CIDR	For more information, see *Network planning* in *VPC User Guide*. **Note** These parameters are available only when you select an **existing VPC**.
Configure SNAT	This parameter is optional. If you clear Configure SNAT for VPC, you must create a NAT gateway or configure SNAT rules for the VPC.
Security Group	You can select **Create Basic Security Group** or **Create Advanced Security Group**. For more information about security groups, see the *Create a security group* topic of *ECS User Guide*.
Access to the Internet	Specify whether to expose the API server with an elastic IP address (EIP). The Kubernetes API server provides multiple HTTP-based RESTful APIs that can be used to create, delete, modify, query, and watch resource objects such as pods and Services. ◦ If you select this check box, an EIP is created and attached to an internal-facing Server Load Balancer (SLB) instance. Port 6443 used by the API server is exposed on the master nodes. You can connect to and manage the cluster by using kubeconfig files over the Internet. ◦ If you clear this check box, no EIP is created. You can connect to and manage the cluster by using kubeconfig files only from within the VPC.

Parameter	Description
SSH Logon	To enable SSH logon, you must first select Expose API Server with EIP. ○ If you enable SSH logon over the Internet, you can access the cluster by using SSH. ○ If you disable SSH logon over the Internet, you cannot access the cluster by using SSH or kubectl. If you want to access an Elastic Compute Service (ECS) instance in the cluster by using SSH, you must manually bind an EIP to the ECS instance and configure security group rules to open SSH port 22.
Ingress	Specify whether to **Install Ingress Controllers**. By default, **Install Ingress Controllers** is selected.
Log Service	If you enable Log Service, you can select an existing project or create a project. If you select **Enable Log Service**, the Log Service plug-in is automatically installed in the cluster. If you select **Create Ingress Dashboard**, Ingress access logs are collected and displayed on dashboards. By default, **Install node-problem-detector and Create Event Center** is selected.
Monitoring Agents	Select or clear **Enable Prometheus Monitoring**. Prometheus Monitoring provides the basic monitoring of the cluster.
Volume Plug-in	By default, **CSI** is selected.
Deletion Protection	If you select this check box, the cluster cannot be deleted in the console or by calling API operations.
Node Protection	This check box is selected by default to prevent nodes from being deleted in the console or by calling API operations.
Labels	Add labels to the cluster.

4. Configure the advanced settings.

Parameter	Description
IP Addresses per Node	The number of IP addresses that can be assigned to a node.

Parameter	Description
Custom Image	You can select a custom image. After you select a custom image, all nodes in the cluster are deployed by using this image.
Kube-proxy Mode	iptables and IPVS are supported. ◦ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the Kubernetes cluster. This mode is suitable for Kubernetes clusters that manage a small number of Services. ◦ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.
Node Port Range	Specify the value of **Node Port Range**.
Taints	Add taints to all worker nodes in the cluster.
Cluster Domain	The default domain name of the cluster is cluster.local. You can specify a custom domain name.
Cluster CA	Specify whether to enable the cluster certification authority (CA) certificate.
User Data	Customize the startup behaviors of ECS instances and import data to the ECS instances. The user data can be used to perform the following operations: ◦ Run scripts during instance startup. ◦ Pass user data as common data into an ECS instance for future reference.

5. Click **Create Cluster** in the upper-right corner of the page.
6. On the **Confirm** page, after all check items are verified, select the terms of service and disclaimer and click **OK** to start the deployment.

   After the cluster is created, you can find the cluster on the **Clusters** page in the console.

5.4. Create a stateless application by using a Deployment

You can deploy a stateless application by using a Deployment. A Deployment can be created by using an image, an orchestration template, or kubectl commands. This topic describes how to create a stateless NGINX application by using an image, an orchestration template, and kubectl commands.

Prerequisites

- A Container Service cluster is created. For more information, see Create a Kubernetes cluster.
- A kubectl client is connected to the cluster. For more information, see Connect to a cluster through kubectl.
- A private image repository is created and your image is uploaded to the repository.

Create a Deployment from an image

Step 1: Configure basic settings

1. Log on to the Container Service console
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**.
5. In the upper part of the **Deployments** page, select the namespace and click **Create from Image**.
6. On the **Basic Information** wizard page, configure the basic settings of the application.

   Before you configure the Deployment, select a namespace in the upper part of the page. In this example, the **default** namespace is selected.

Parameter	Description
Name	Enter a name for the application.
Replicas	The number of pods that you want to provision for the application. Default value: 2.
Type	The type of the application. You can select **Deployment**, **StatefulSet**, **Job**, **CronJob**, or **DaemonSet**.
Label	Add labels to the application. The labels are used to identify the application.
Annotations	Add annotations to the application.
Synchronize Timezone	Specify whether to synchronize the time zone between nodes and containers.

   > **Note** In this example, the **Deployment** type is selected.

7. Click **Next**.

Proceed to the **Container** wizard page.

Step 2: Configure containers

On the **Container** wizard page, configure the following container settings: the container image, resource request and limit, container ports, environment variables, health check settings, lifecycle configurations, volumes, and logging configurations.

> **Note** On the **Container** wizard page, Click **Add Container** to add more containers to the pod.

1. In the **General** section, configure the basic settings of the container.

Parameter	Description
Image	To select an image, click **Select Image**. In the upper right corner of the Select Image dialog box, select **Default** or **enterprise-edition** from the **Container Registry Instance** drop-down list. Then, select the image that you want to use and click **OK**. In this example, the nginx image is selected. **Note** ○ Default: shows images that are stored on the Container Registry Standard Edition instance. ○ enterprise-edition: shows images that are stored on the Container Registry Advanced Edition instance. You can also enter the address of a private image registry. Specify a private registry in the following format: `domainname/namespace/imagename`.
Image Version	○ Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used. ○ You can select the following image pulling policies: ■ **ifNotPresent**: If the image that you want to pull is found on your on-premises machine, the image on your on-premises machine is used. Otherwise, Container Service pulls the image from the image registry. ■ **Always**: Container Service pulls the image from the registry each time the application is deployed or expanded. ■ **Never**: Container Service uses only images on your on-premise machine. **Note** If you select **Image Pull Policy**, no image pulling policy is applied. ○ To pull the image without a password, click **Set Image Pull Secret** to configure a Secret for pulling images.
Resource Limit	You can specify an upper limit for the CPU, memory, and ephemeral storage space that the container can consume. This prevents the container from occupying an excessive amount of resources.

Parameter	Description
Required Resources	The amount of the CPU resources, memory resources, and ephemeral storage space that are reserved for this application. These resources are exclusive to the container. This prevents the application from becoming unavailable when other services or processes compete for computing resources.
Container Start Parameter	◦ stdin: specifies that start parameters are sent to the container as standard input (stdin). ◦ tty: specifies that start parameters defined in a virtual terminal are sent to the container. The two options are usually used together. In this case, the virtual terminal (tty) is associated to the stdin of the container. For example, an interactive program receives the stdin from the user and displays the content in the terminal.
Privileged Container	◦ If you select Privileged Container, privileged=true is specified for the container and the privilege mode is enabled. ◦ If you do not select Privileged Container, privileged=false is specified for the container and the privilege mode is disabled.
Init Container	If you select Init Container, an init container is created. An init container provides tools for pod management. For more information, see init containers.

2. (Optional)In the **Ports** section, click **Add** to configure one or more container ports.

Parameter	Description
Name	Enter a name for the port.
Container Port	The container port that you want to open. Valid values: 1 to 65535.
Protocol	Valid values: TCP and UDP.

3. (Optional)In the **Environments** section, click **Add** to set environment variables.

   You can configure environment variables in key-value pairs for pods. Environment variables are used to apply pod configurations to containers. For more information, see Pod variables.

Parameter	Description

Parameter	Description
Type	Select the type of environment variable. Valid values: ○ **Custom** ○ **ConfigMaps** ○ **Secret** ○ **Value/ValueFrom** ○ **ResourceFieldRef**
Variable	Specify the name of the environment variable.
Value/ValueFrom	Specify a reference that is used to define the environment variable.

4. (Optional)In the **Health Check** section, you can enable **liveness** and **readiness** probes as needed.

   For more information, see Configure Liveness, Readiness and Startup Probes.

Parameter	Request type	Configuration description

Parameter	Request type	Configuration description
	HTTP	Sends an HTTP GET request to the container. You can set the following parameters: ◦ Protocol: HTTP or HTTPS. ◦ Path: the requested path on the server. ◦ Port: Enter the container port that you want to open. Valid values: 1 to 65535. ◦ HTTP Header: the custom headers in the HTTP request. Duplicate headers are allowed. You can set HTTP headers in key-value pairs. ◦ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the wait time (in seconds) before the first probe is performed after the container is started. Default value: 3. ◦ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ◦ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ◦ Healthy Threshold: the minimum number of consecutive successes that must occur before a container is considered healthy after a failed probe. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ◦ Unhealthy Threshold: the minimum number of consecutive failures that must occur before a container is considered unhealthy after a success. Default value: 3. Minimum value: 1.

Parameter	Request type	Configuration description
○ Liveness: Liveness probes are used to determine when to restart the container. ○ Readiness: Readiness probes are used to determine whether the container is ready to accept traffic.	TCP	Sends a TCP socket to the container. kubelet attempts to open the socket on the specified port. If the connection can be established, the container is considered healthy. Otherwise, the container is considered unhealthy. You can configure the following parameters: ○ Port: Enter the container port that you want to open. Valid values: 1 to 65535. ○ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the wait time (in seconds) before the first probe is performed after the container is started. Default value: 15. ○ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ○ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ○ Healthy Threshold: the minimum number of consecutive successes that must occur before a container is considered healthy after a failed probe. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ○ Unhealthy Threshold: the minimum number of consecutive failures that must occur before a container is considered unhealthy after a success. Default value: 3. Minimum value: 1.

Parameter	Request type	Configuration description
	Command	Runs a probe command in the container to check the health status of the container. You can configure the following parameters: ◦ Command: the probe command that is run to check the health status of the container. ◦ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the wait time (in seconds) before the first probe is performed after the container is started. Default value: 5. ◦ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ◦ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ◦ Healthy Threshold: the minimum number of consecutive successes that must occur before a container is considered healthy after a failed probe. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ◦ Unhealthy Threshold: the minimum number of consecutive failures that must occur before a container is considered unhealthy after a success. Default value: 3. Minimum value: 1.

5. (Optional)In the **Lifecycle** section, configure the lifecycle of the container.

   You can set the following parameters to configure the lifecycle of the container: Start, Post Start, and Pre Stop. For more information, see Configure the lifecycle of a container.

Parameter	Description
Start	Specify a command and parameter that take effect before the container starts.
Post Start	Specify a command that takes effect after the container starts.
Pre Stop	Specify a command that takes effect before the container stops.

6. (Optional)In the **Volume** section, you can mount local volumes and persistent volume claims (PVCs) to the container.

Parameter	Description
Add Local Storage	You can select HostPath, ConfigMap, Secret, or EmptyDir from the PV Type drop-down list. Then, set Mount Source and Container Path to mount the volume to the container. For more information, see Volumes.
Add PVC	You can mount persistent volumes (PVs) by using PVCs. You must create a PVC before you can select the PVC to mount a PV. For more information, see Create a PVC.

7. (Optional)In the **Log** section, you can specify logging configurations and add custom tags to the collected log.

   Notice Make sure that the Log Service agent is installed in the cluster.

Parameter	Description
Collection Configuration	Logstore: creates a Logstore in Log Service to store the collected log.
	Log Path in Container: Specify stdout or a container path to collect log data. ○ Collect stdout files: If you specify stdout, the stdout files are collected. ○ Text Logs: specifies that the log in the specified path of the container is collected. In this example, */var/log/nginx* is specified as the path. Wildcard characters can be used in the path.
Custom Tag	You can also add custom tags. The tags are added to the log of the container when the log is collected. Custom tags provide an easy method to filter collected the log and perform statistical analysis.

8. Click **Next**.

   Proceed to the **Advanced** wizard page.

Step 3: Configure advanced settings

On the **Advanced** wizard page, configure the following settings: access control, scaling, scheduling, annotations, and labels.

1. In the **Access Control** section, you can configure access control settings for exposing backend pods.

> **Note**
>
> You can configure the following access control settings based on your business requirements:
>
> - Internal applications: For applications that provide services within the cluster, you can create a **ClusterIP** or **NodePort** Service to enable internal communication.
> - External applications: For applications that are exposed to the Internet, you can configure access control by using one of the following methods:
>   - Create a LoadBalancer Service. When you create a Service, set Type to **Server Load Balancer**. You can select or create a Server Load Balancer (SLB) instance for the Service and use the Service to expose your application to the Internet.
>   - Create an Ingress and use it to expose your application to the Internet. For more information, see Ingress.

You can also specify how the backend pods are exposed to the Internet. In this example, a ClusterIP Service and an Ingress are created to expose the NGINX application to the Internet.

- Click **Create** to the right side of **Services**. In the Create Service dialog box, configure the following parameters.

Parameter	Description
Name	Enter a name for the Service.

Parameter	Description
Type	The type of Service. This parameter determines how the Service is accessed. ■ **Cluster IP**: The ClusterIP type Service. This type of Service exposes the Service by using an internal IP address of the cluster. If you select this type, the Service is accessible only within the cluster. This is the default type. **Note** The **Headless Service** check box is displayed only when you set Type to **Cluster IP**. ■ **Node Port**: The NodePort type Service. This type of Service is accessed by using the IP address and a static port of each node. A NodePort Service can be used to route requests to a ClusterIP Service. The ClusterIP Service is automatically created by the system. You can access a NodePort Service from outside the cluster by sending requests to `<NodeIP>:<NodePort>`. ■ **Server Load Balancer**: The LoadBalancer type Service. This type of Service exposes the Service by using an SLB instance. If you select this type, you can enable internal or external access to the Service. SLB instances can be used to route requests to NodePort and ClusterIP Services. ■ Create SLB Instance: You can click **Modify** to change the specification of the SLB instance. ■ Use Existing SLB Instance: You can select an existing SLB instance. **Note** You can create an SLB instance or use an existing SLB instance. You can also associate an SLB instance with more than one Service. However, you must take note of the following limits: ■ If you use an existing SLB instance, the listeners of the SLB instance overwrite the listeners of the Service. ■ If a Classic Load Balancer (CLB) instance is created along with a Service, you cannot reuse this CLB instance when you create other Services. Otherwise, the CLB instance may be deleted. Only CLB instances that are manually created in the console or by calling the API can be used to expose multiple Services. ■ Kubernetes services that share the same CLB instance must use different frontend listening ports. Otherwise, port conflicts may occur. ■ If multiple Kubernetes Services share the same SLB instance, you must use the listener names and the vServer group names as unique identifiers in Kubernetes. Do not modify the names of listeners or vServer groups. ■ You cannot share SLB instances across clusters.

Parameter	Description
Port Mapping	Specify a Service port and a container port. The container port must be the same as the one that is exposed in the backend pod.
External Traffic Policy	■ Local: Traffic is routed to only the node where the Service is deployed. ■ Cluster: Traffic can be routed to pods on other nodes. **Note** The **External Traffic Policy** parameter is available only if you set Type to **Node Port** or **Server Load Balancer**.
Annotations	Add one or more annotations to the Service to configure the SLB instance. For example, `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20` specifies that the maximum bandwidth of the Service is 20 Mbit/s. This limits the amount of traffic that flows through the Service.
Label	Add one or more labels to the Service. Labels are used to identify the Service.

- To create an Ingress, click **Create** to the right side of **Ingresses**. In the Create dialog box, set the parameters.

  For more information about the parameters that are required for creating an Ingress, see Ingress configurations.

  Notice When you create an application from an image, you can create an Ingress only for one Service. In this example, a virtual hostname is specified as the test domain name. You must add a mapping to the *hosts* file for this domain name in the following format: *External endpoint of the Ingress* + *domain name of the Ingress*. In actual scenarios, use a domain name that has obtained an Internet Content Provider (ICP) number.

  ```
  101.37.xx.xx    foo.bar.com    # The IP address of the Ingress.
  ```

Parameter	Description
Name	The name of the Ingress.

Parameter	Description
Rules	Ingress rules are used to enable access to specified Services in a cluster. For more information, see Ingress configurations. ■ **Domain**: Enter the domain name of the Ingress. ■ **Path**: Enter the Service URL. The default path is the root path /. The default path is used in this example. Each path is associated with a backend Service. SLB forwards traffic to a backend Service only when inbound requests match the domain name and path. ■ **Services**: Select a Service and a Service port. ■ **EnableTLS**: Select this check box to enable TLS.
Weight	Set the weight for each Service in the path. Each weight is calculated as a percentage value. Default value: 100.
Canary Release	After a canary release rule is configured, only requests that match the rule are routed to the Service of the new application version. If the weight of new-nginx is lower than 100%, requests that match the specified rules are routed to the Service based on the Service weight. Container Service supports multiple traffic splitting methods. This allows you to select suitable solutions for specific scenarios, such as canary releases and A/B testing: ■ Traffic splitting based on request headers ■ Traffic splitting based on cookies ■ Traffic splitting based on query parameters **Note** Only Ingress controllers of 0.12.0-5 and later versions support traffic splitting. The following parameters are required: ■ **Services**: Specify the Services to be accessed. ■ **Type**: Select the type of matching rule. Valid values: Header, Cookie, and Query. ■ **Name** and **Match Value**: user-defined request parameters that are specified in key-value pairs. ■ **Matching Rule**: Regular expressions and exact matches are supported.

Parameter	Description
Annotations	Click the + icon to add an annotation. You can select **Custom Annotation** or **Ingress-NGINX** from the Type drop-down list. For more information about Ingress annotations, see Annotations. **Note** If you want to add an annotation for configuring rewrite rules, select **Ingress-NGINX** from the Type drop-down list. Select **nginx.ingress.kubernetes.io/rewrite-target Rewrite Annotation** from the **Name** drop-down list. Then, specify / in the **Value** column. This annotation specifies that the requests destined for */path* are redirected to the root path, which can be recognized by the backend Service.
Labels	Add labels to describe the characteristics of the Ingress.

You can find the created Service and Ingress in the **Access Control** section. You can click **Update** or **Delete** to change the configurations.

2. (Optional)In the **Scaling** section, specify whether to enable **HPA**. Horizontal Pad Autoscaler (HPA) allows you to meet the resource requirements of the application at different load levels.

   HPA can automatically scale the number of pods in a Container Service cluster based on the CPU and memory usage.

   Note To enable HPA, you must configure resources required by the container. Otherwise, HPA does not take effect.

Parameter	Description
Metric	Select CPU Usage or Memory Usage. The selected resource type must be the same as that specified in the Required Resources field.
Conditions	Specify the resource usage threshold. HPA triggers scale-out events when the threshold is exceeded.
Max. Replica	Specify the maximum number of pods to which the application can be scaled.
Min. Replica	Specify the minimum number of pods that must run.

3. (Optional)In the **Scheduling** section, you can configure the following parameters: **Update**

Method, **Node Affinity**, **Pod Affinity**, **Pod Anti Affinity**, and **Toleration**. For more information, see Affinity and anti-affinity.

> **Note** Node affinity and pod affinity affect pod scheduling based on node labels and pod labels. You can add node labels and pod labels that are provided by Kubernetes to configure node affinity and pod affinity. You can also add custom labels to nodes and pods, and then configure node affinity and pod affinity based on these custom labels.

Parameter	Description
Update Method	Select Rolling Update or OnDelete. For more information, see Deployments.
Node Affinity	Set **Node Affinity** by selecting worker node labels as selectors. Node affinity supports required and preferred rules, and various operators such as In, NotIn, Exists, DoesNotExist, Gt, and Lt. ○ **Required**: Specify the rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution affinity. These rules have the same effect as the `NodeSelector` parameter. In this example, pods can be scheduled only to nodes with the specified labels. You can create multiple required rules. However, only one of them must be met. ○ **Preferred**: Specify the rules that are not required to be matched for pod scheduling. In the YAML file, these rules are defined by the preferredDuringSchedulingIgnoredDuringExecution affinity. If you specify a preferred rule, the scheduler attempts to schedule a pod to a node that matches the preferred rule. You can also set weights for preferred rules. If multiple nodes match the rule, the node with the highest weight is preferred. You can create multiple preferred rules. However, all of them must be met before the pod can be scheduled.

Parameter	Description
Pod Affinity	Pod affinity rules specify how pods are deployed relative to other pods in the same topology domain. For example, you can use pod affinity to deploy services that communicate with each other to the same topological domain, such as a host. This reduces the network latency between these services. Pod affinity enables you to specify to which node pods can be scheduled based on the labels of running pods. Pod affinity supports required and preferred rules, and the following operators: `In, NotIn, Exists, and DoesNotExist`. ○ **Required**: Specify rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution affinity. A node must match the required rules before pods can be scheduled to the node. ■ **Namespace**: Specify the namespace to apply the required rule. Pod affinity rules are defined based on the labels that are added to pods and therefore must be scoped to a namespace. ■ **Topological Domain**: Set the topologyKey. This specifies the key for the node label that the system uses to denote the topological domain. For example, if you set the parameter to `kubernetes.io/hostname`, topologies are determined by nodes. If you set the parameter to `beta.kubernetes.io/os`, topologies are determined by the operating systems of nodes. ■ **Selector**: Click the plus icon to the right of Selector. You can specify multiple selectors of the rule. You can set the Label Name, Operator, Label Value parameters for each selector. In this example, the required rule specifies that the application to be created is scheduled to a host that runs applications with the `app:nginx` label. ■ **View Applications**: Click **View Applications** and set the namespace and application in the dialog box that appears. You can view the pod labels of the selected application and add these labels as selectors. ○ **Preferred**: Specify the rules that are not required to be matched for pod scheduling. In the YAML file, these rules are defined by the preferredDuringSchedulingIgnoredDuringExecution affinity. If you specify a preferred rule, the scheduler attempts to schedule the pod to a node that matches the preferred rule. You can set weights for preferred rules. The other parameters are the same as those of required rules. **Note** **Weight**: Set the weight of a preferred rule to a value from 1 to 100. The scheduler calculates the weight of each node that meets the preferred rule based on an algorithm, and then schedules the pod to the node with the highest weight.

Parameter	Description
Pod Anti Affinity	Pod anti-affinity rules specify that pods are not scheduled to topological domains where pods with matching labels are deployed. Pod anti-affinity rules apply to the following scenarios: ○ Schedule the pods of an application to different topological domains, such as multiple hosts. This allows you to enhance the stability of your service. ○ Grant a pod exclusive access to a node. This enables resource isolation and ensures that no other pods can share the resources of the specified node. ○ Schedule pods of an application to different hosts if the pods may interfere with each other. **Note** The parameters of pod anti-affinity rules are the same as those of pod affinity rules. Create rules based on scenarios.
Toleration	Configure toleration rules to allow pods to be scheduled to nodes with matching taints.

4. (Optional)In the **Labels,Annotations** section, click **Add** to configure labels and annotations for the pod.

Parameter	Description
Pod Labels	Add a label to the pod. The label is used to identify the application.
Pod Annotations	Add an annotation to the pod configurations.

5. Click **Create**.

Step 4: Check the application

1. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments** . On the Deployments page, you can find that the application that you created is displayed.
2. In the left-side navigation pane of the cluster details page, choose **Network > Ingresses** . On the Ingresses page, you can find that the Ingress that you created is displayed.
3. Enter the test domain name in the address bar of your browser and press Enter. The NGINX welcome page appears.

What to do next

View application details

In the left-side navigation pane of the Container Service console, click **Clusters**. Click the name of the cluster where the application is deployed or click **Details** in the **Actions** column. Choose **Workloads > Deployments**. On the **Deployments** page, click the name of the deployed application or click **Details** in the **Actions** column.

On the details page of the application, you can view the YAML file of the application. You can also edit, scale, redeploy, and refresh the application.

Operation	Description
Edit	On the details page of the application, click **Edit** in the upper-right corner of the page to modify the application configurations.
Scale	On the details page of the application, click **Scale** in the upper-right corner of the page to scale the application to a required number of pods.
View in YAML	On the details page of the application, click **View in YAML** to **update** or **download** the YAML file. You can also click **Save As** to save the file as a different name.
Redeploy	On the details page of the application, click **Redeploy** in the upper-right corner of the page to redeploy the application.
Refresh	On the details page of the application, click **Refresh** in the upper-right corner of the page to refresh the details page.

On the details page of the application, you can view information about the pods, pod events, historical versions of the application, application log, access methods of the application, and application triggers.

Item	Description
Pods	Click the **Pods** tab to view information about the pods that are provisioned for the application.
Access methods	Click the **Access Methods** tab to view information about the access methods of the application. You can also perform the following operations: • Click **Create** to the right side of **Services** to create a Service. For more information about the parameters required for creating a Service, see Create a service. • Click **Create** to the right side of **Ingresses** to create an Ingress. For more information about the parameters required for creating an Ingress, see Ingress configurations.
Pod events	Click the **Events** tab to view information about the pod events.

Item	Description
Pod scaling	Click the **Pod Scaling** tab to configure HPA to scale the application based on the CPU utilization and memory utilization of the application. Click **Create** to the right side of **HPA** to configure HPA. For more information about the parameters required for configuring HPA, see Metric scaling.
Historical application versions	Click the **History Versions** tab to view the historical versions of the application.
Application log	Click the **Logs** tab to view the application log.
Triggers	On the details page of the application, click the **Triggers** tab to create application triggers. For more information about application triggers, see Create a trigger on an application.

Manage the application

On the **Deployments** page, select the application and click **More** in the **Actions** column to perform the following operations.

Operation	Description
View in YAML	View the YAML file of the application.
Redeploy	Redeploy the application.
Edit Label	Configure the labels of the application.
Edit Annotations	Configure the annotations of the application.
Node Affinity	Configure the node affinity settings of the application. For more information, see Scheduling.
Scaling	Configure the scaling settings of the application. For more information, see Horizontal pod autoscaling.
Toleration	Configure the toleration rules of the application. For more information, see Scheduling.
Upgrade Policy	Configure the update policy of the application. • **Rolling Update**: Pods are updated in a rolling update fashion. • **OnDelete**: All existing pods are deleted before new pods are created.
Clone	Create a new application by using the same container settings as the current application.
Roll Back	Roll back the application to a previous version.
Logs	View the log of the application.

Operation	Description
Delete	Delete the application.

Create a Linux application by using an orchestration template

In an orchestration template, you must define the resource objects that are required for running an application and configure mechanisms such as label selectors to manage the resource objects that make up the application.

This section describes how to use an orchestration template to create an NGINX application that consists of a Deployment and a Service. The Deployment provisions pods for the application and the Service manages access to the backend pods.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**.
5. In the upper part of the **Deployments** page, select a namespace and click **Create from Template**.
6. Set the parameters and click **Create**.
   - **Sample Template**: Container Service provides YAML templates of various resource types. This simplifies the deployment of resource objects. You can also create a custom template based on the YAML syntax to define the resources that you want to create.
   - **Add Deployment**: This feature allows you to define a YAML template.
   - **Use Existing Template**: You can import an existing template to the configuration page.
   - **Save As**: You can save the template that you have configured.

   The following sample template is based on an orchestration template provided by Container Service. You can use this template to create a Deployment to run an NGINX application.
   - Container Service supports the YAML syntax. You can use the `---` symbol to separate multiple resource objects. This allows you to create multiple resource objects in a single template.
   - (Optional)By default, if you mount a volume to the application, the existing files in the mount path of the application are overwritten. To avoid the existing files from being overwritten, you can add a `subPath` parameter.

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
    name: nginx-deployment
    labels:
      app: nginx
spec:
    replicas: 2
    selector:
      matchLabels:
        app: nginx
    template:
      metadata:
        labels:
          app: nginx
      spec:
        containers:
        - name: nginx
          image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
          ports:
          - containerPort: 80
---
apiVersion: v1     # for versions before 1.8.0 use apps/v1beta1
kind: Service
metadata:
   name: my-service1        #TODO: to specify your service name
   labels:
     app: nginx
spec:
   selector:
     app: nginx             #TODO: change label selector to match your backend pod
   ports:
   - protocol: TCP
     name: http
     port: 30080            #TODO: choose an unique port on each node to avoid port con
flict
     targetPort: 80
   type: LoadBalancer        ##In this example, the type is changed from NodePort to Lo
adBalancer.
```

7. Click **Create**. A message that indicates the deployment status appears.

Manage applications by using commands

This topic describes how to create an application or view containers of an application by using commands.

Create an application by using commands

1. In a CLI, run the following command to start a container. An NGINX web server is used in this example.

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

2. Run the following command to create a service entry for this container. The `--type=LoadBalancer`

parameter in the command indicates that the system must create a Server Load Balancer (SLB) instance for the NGINX container.

```
kubectl expose deployment nginx --port=80 --target-port=80 --type=LoadBalancer
```

View containers by using commands

In the CLI, run the `kubectl get pods` command to list all running containers in the default namespace.

```
NAME                                READY     STATUS    RESTARTS   AGE
nginx-2721357637-d****              1/1       Running   1          9h
```

Create an application by using an image pull Secret

Container Service allows you to use image pull Secrets in the Container Service console. You can create an image pull Secret or use an existing image pull Secret.

When you create an application from a private image, you must set a Secret for image pulling to ensure the security of the image. In the Container Service console, you can specify the authentication information of the private image repository as a Secret of the docker-registry type. This Secret applies to the specified Kubernetes cluster.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, select the namespace and click **Create from Image** in the upper-right corner of the page.
6. On the **Basic Information** wizard page, set the parameters. For more information, see Configure basic settings.
7. Configure containers.

   The following example describes how to configure an image pull Secret. For more information about container configurations, see Basic container configurations.

   i. On the **Container** wizard page, enter a private image registry in the **Image Name** field. The private image registry must be in the following format: `domainname/namespace/imagename`.

   > **Note** You do not need to configure a image pulling Secret for a public image registry.

   ii. Enter the image version that you want to use in the Image Version field.

iii. Click **Set Image Pull Secret**, and create a Secret or select an existing Secret in the dialog box that appears.

- You can select **New Secret**. If you select New Secret, set the following parameters:

Parameter	Description
Name	The key name of the Secret. You can enter a custom name.
Repository Domain	The address of the specified Docker repository. If you use an image repository in Container Registry, the address is automatically specified.
Username	The username that is used to log on to the Docker repository. If you use an image repository in Container Registry, the username is the username of your Alibaba Cloud account.
Password	The password used to log on to the Docker repository. If you use an image repository in Container Registry, the password is the logon password of the image repository in Container Registry.
Email	The email address. This parameter is optional.

  Click **OK**. The Secret appears on the page after it is created.

- You can also select **Existing Secret**. You can use commands or YAML files to create image pulling Secrets.

8. Configure other parameters based on your requirements. Then, click **Create**.

   For more information, see Step 3: Configure advanced settings.

9. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**. On the **Deployments** page, check the status of the application. If the application is running as normal, it indicates that the image is successfully pulled by using the Secret that you configured.

6.Kubernetes clusters

6.1. Authorizations

6.1.1. Assign RBAC roles to a RAM user

This topic describes how to assign role-based access control (RBAC) roles to Resource Access Management (RAM) users. By default, RBAC is enabled for Kubernetes 1.6 and later. RBAC is important for you to manage clusters. You can use RBAC to specify the types of operations that are allowed for specific users based on their roles in an organization.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Authorizations**.
3. On the **RAM Users** tab, select the RAM user to which you want to grant permissions and click **Modify Permissions**.

   > **Note** If you log on to the Container Service console as a RAM user, make sure that the RAM user is assigned the predefined RBAC administrator role or the cluster-admin role.

4. On the **Configure Role-Based Access Control (RBAC)** wizard page, click **Add Permissions** to add cluster-scoped or namespace-scoped permissions and select a predefined or custom RBAC role in the Permission column. You can also click the minus sign (-) to delete permissions. After you add the permissions, click **Next Step**.

   > **Note** For each RAM user, you can assign only one predefined RBAC role but one or more custom RBAC roles to manage the same cluster or namespace.

   The following table describes the permissions that the predefined and custom RBAC roles have on clusters and namespaces.

Roles and permissions

Role	RBAC permissions on cluster resources
Administrator	Read and write permissions on resources in all namespaces.
O&M Engineer	Read and write permissions on resources in all namespaces and read-only permissions on nodes, persistent volumes (PVs), namespaces, and service quotas within a cluster.
Developer	Read and write permissions on resources in a specified namespace or all namespaces.
Restricted User	Read-only permissions on resources in a specified namespace or all namespaces.

Role	RBAC permissions on cluster resources
Custom	The cluster role that you select for a custom role determines what permissions the custom role has. Before you select a cluster role, make sure that you are aware of the permissions that the cluster role has in case the RAM user is granted excessive permissions.

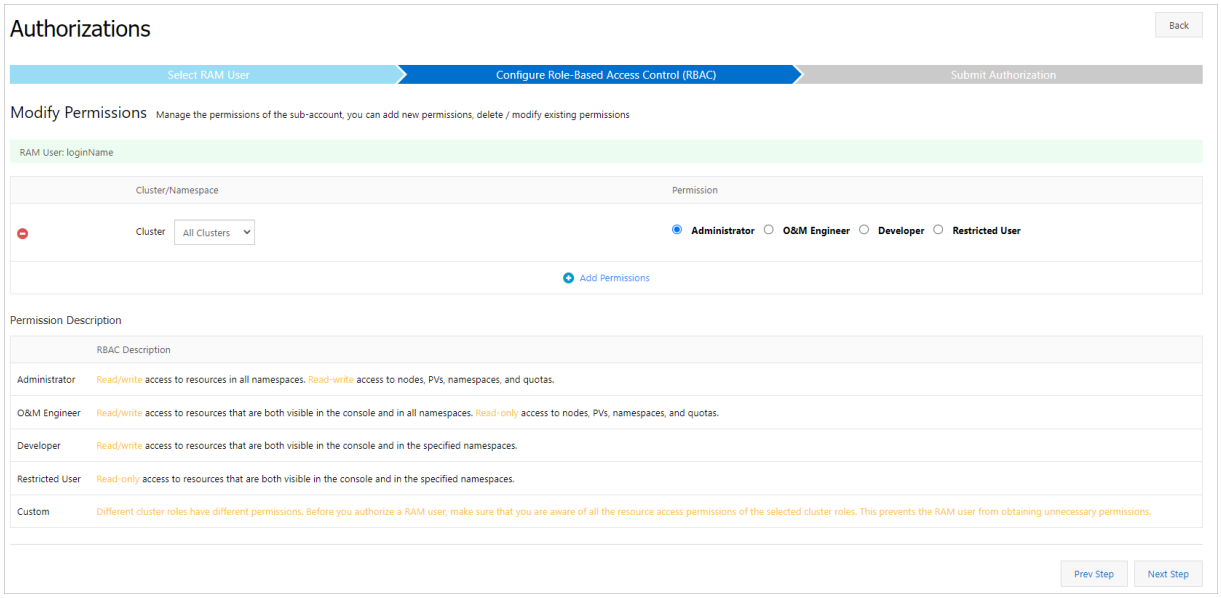
After the authorization is complete, you can use the account of the specified RAM user to log on to the Container Service console. For more information, see Log on to the Container Service console.

Predefined and custom RBAC roles

Container Service provides the following predefined RBAC roles: administrator, O&M engineer, developer, and restricted user. You can use these roles to regulate Container Service access control in most scenarios. In addition, you can use custom roles to customize permissions on clusters.

Container Service provides a set of custom RBAC roles.

> **Note** The cluster-admin role is similar to a super administrator. By default, the cluster-admin role has the permissions to manage all resources within a cluster.



You can log on to a master node of a cluster and run the following command to view the custom RBAC roles that are assigned to the current account:

```
# kubectl get clusterrole
```

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

Run the following command to view the details of a role, for example, the cluster-admin role:

```
# kubectl get clusterrole cluster-admin -o yaml
```

Notice After a RAM user is assigned the cluster-admin role, the RAM user has the same permissions as the Alibaba Cloud account to which the RAM user belongs. The RAM user has full control over all resources within the cluster. Proceed with caution.

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

6.2. Clusters

6.2.1. Create a Kubernetes cluster

This topic describes how to create a Kubernetes cluster in the Container Service console.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**. On the Clusters page, click **Create Kubernetes Cluster** in the upper-right corner.
3. On the **Dedicated Kubernetes** tab of the **Create Cluster** page, set the following parameters.

Parameter	Description
Cluster Name	Enter a name for the cluster. The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-). **Note** The cluster name must be unique among clusters that belong to the same Alibaba Cloud account.
Resource Set	Select a resource set. A resource set is a container used to store resources. Each resource must belong to a resource set. After you select a resource set, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource set.
Region	Select the region where you want to deploy the cluster.
VPC	You can select a VPC from the drop-down list. ○ If the specified VPC has a NAT gateway, Container Service uses this NAT gateway. ○ If the VPC does not have a NAT gateway, the system automatically creates one. If you do not want the system to create a NAT gateway, clear **Configure SNAT for VPC**. **Note** If you disallow the system to automatically create a NAT gateway and want the VPC to access the Internet, you must manually associate the VPC with a NAT gateway or create SNAT rules for the VPC.

Parameter	Description
vSwitch	Select vSwitches. You can select up to three vSwitches that are deployed in different **zones**.
Kubernetes Version	Select a Kubernetes version.
Container Runtime	You can select Docker or Sandboxed-Container.
Master Configurations	Set the Instance Type and System Disk parameters: ○ Master Node Quantity: You can add up to three master nodes. ○ Instance Type: You can select multiple instance types. For more information, see *Instance families and instance types* in the *ECS* documentation. ○ System Disk: **SSD Disk**, **Ultra Disk**, and **Enhanced SSD** are supported. **Note** You can select **Enable Backup** to back up disk data.
Worker Instance	You can select **Create Instance** or **Add Existing Instance**.

Parameter	Description
Worker Configurations	If **Worker Instance** is set to **Create Instance**, set the following parameters: ○ Instance Type: You can select multiple instance types. For more information, see *Instance families and instance types* in the *ECS* documentation. ○ Selected Types: The selected instance types are displayed. ○ Quantity: Set the number of worker nodes. ○ System Disk: **SSD Disk**, **Ultra Disk**, and **Enhanced SSD** are supported. Note You can select **Enable Backup** to back up disk data. ○ Mount Data Disk: **SSD Disk**, **Ultra Disk**, and **Enhanced SSD** are supported. Note ■ You can select **Encrypt Disk** to encrypt disks. ■ You can select **Enable Backup** to back up disk data.
Operating System	The CentOS and Alibaba Cloud Linux operating systems are supported.
Password	Set a password that is used to log on to the nodes. Note The password must be 8 to 30 characters in length, and must contain at least three of the following types of character: uppercase letters, lowercase letters, digits, and special characters.
Confirm Password	Enter the password again.
Network Plug-in	Flannel and Terway are supported. By default, Flannel is selected.

Parameter	Description
Pod CIDR Block and Service CIDR	For more information, see *Network planning* in *VPC User Guide*. **Note** These parameters are available only when you select an **existing VPC**.
Configure SNAT	This parameter is optional. If you clear Configure SNAT for VPC, you must create a NAT gateway or configure SNAT rules for the VPC.
Security Group	You can select **Create Basic Security Group** or **Create Advanced Security Group**. For more information about security groups, see the *Create a security group* topic of *ECS User Guide*.
Access to the Internet	Specify whether to expose the API server with an elastic IP address (EIP). The Kubernetes API server provides multiple HTTP-based RESTful APIs that can be used to create, delete, modify, query, and watch resource objects such as pods and Services. ○ If you select this check box, an EIP is created and attached to an internal-facing Server Load Balancer (SLB) instance. Port 6443 used by the API server is exposed on the master nodes. You can connect to and manage the cluster by using kubeconfig files over the Internet. ○ If you clear this check box, no EIP is created. You can connect to and manage the cluster by using kubeconfig files only from within the VPC.
SSH Logon	To enable SSH logon, you must first select Expose API Server with EIP. ○ If you enable SSH logon over the Internet, you can access the cluster by using SSH. ○ If you disable SSH logon over the Internet, you cannot access the cluster by using SSH or kubectl. If you want to access an Elastic Compute Service (ECS) instance in the cluster by using SSH, you must manually bind an EIP to the ECS instance and configure security group rules to open SSH port 22.
Ingress	Specify whether to **Install Ingress Controllers**. By default, **Install Ingress Controllers** is selected.

Parameter	Description
Log Service	If you enable Log Service, you can select an existing project or create a project. If you select **Enable Log Service**, the Log Service plug-in is automatically installed in the cluster. If you select **Create Ingress Dashboard**, Ingress access logs are collected and displayed on dashboards. By default, **Install node-problem-detector and Create Event Center** is selected.
Monitoring Agents	Select or clear **Enable Prometheus Monitoring**. Prometheus Monitoring provides the basic monitoring of the cluster.
Volume Plug-in	By default, **CSI** is selected.
Deletion Protection	If you select this check box, the cluster cannot be deleted in the console or by calling API operations.
Node Protection	This check box is selected by default to prevent nodes from being deleted in the console or by calling API operations.
Labels	Add labels to the cluster.

4. Configure the advanced settings.

Parameter	Description
IP Addresses per Node	The number of IP addresses that can be assigned to a node.
Custom Image	You can select a custom image. After you select a custom image, all nodes in the cluster are deployed by using this image.

Parameter	Description
Kube-proxy Mode	iptables and IPVS are supported. ○ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the Kubernetes cluster. This mode is suitable for Kubernetes clusters that manage a small number of Services. ○ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.
Node Port Range	Specify the value of **Node Port Range**.
Taints	Add taints to all worker nodes in the cluster.
Cluster Domain	The default domain name of the cluster is cluster.local. You can specify a custom domain name.
Cluster CA	Specify whether to enable the cluster certification authority (CA) certificate.
User Data	Customize the startup behaviors of ECS instances and import data to the ECS instances. The user data can be used to perform the following operations: ○ Run scripts during instance startup. ○ Pass user data as common data into an ECS instance for future reference.

5. Click **Create Cluster** in the upper-right corner of the page.
6. On the **Confirm** page, after all check items are verified, select the terms of service and disclaimer and click **OK** to start the deployment.

   After the cluster is created, you can find the cluster on the **Clusters** page in the console.

6.2.2. Create a Kubernetes cluster by using a custom image

In the cloud-native era, an increasing number of users choose to migrate applications and businesses to the cloud. The requirement on the container platform varies based on different business scenarios. To meet business requirements, many users want to create Kubernetes clusters by using custom images. This topic describes how to create a Kubernetes cluster by using a custom image.

Prerequisites

Before you create a Kubernetes cluster by using a custom image, take note of the following limits on the custom images that are supported by Container Service:

We recommend that you use the latest base images provided by Container Service. The base images of Container Service can be used to create Kubernetes clusters and have passed the strict tests of the Container Service technical team. Custom images that are used to create Kubernetes clusters must meet the following requirements:

- Alibaba Cloud cloud-init can be installed. For more information, see the *Import custom images* chapter of *ECS User Guide*.
- If you want to create a dedicated Kubernetes cluster by using a custom image, you must enable the sshd server and use the default port 22. This allows you to transfer files to cluster nodes after the nodes are enabled. For more information, see Connect to a master node by using SSH.
- Time synchronization is performed by using a Network Time Protocol (NTP) server provided by Alibaba Cloud.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**. On the Clusters page, click **Create Kubernetes Cluster** in the upper-right corner.
3. On the **Dedicated Kubernetes** tab of the **Create Cluster** page, click **Select** to the right side of **Custom Image**. In the **Choose Custom Image** dialog box, click **Use** to the right side of the image that you want to use. For more information about other parameters, see Create a Kubernetes cluster.
4. Click **Create Cluster** in the upper-right corner of the page.
5. On the **Confirm** page, after all check items are verified, select the terms of service and disclaimer and click **OK** to start the deployment.

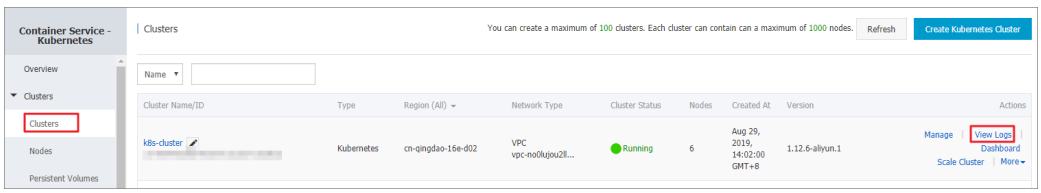
   After the cluster is created, you can find the cluster on the **Clusters** page in the console.

6.2.3. View log files of a cluster

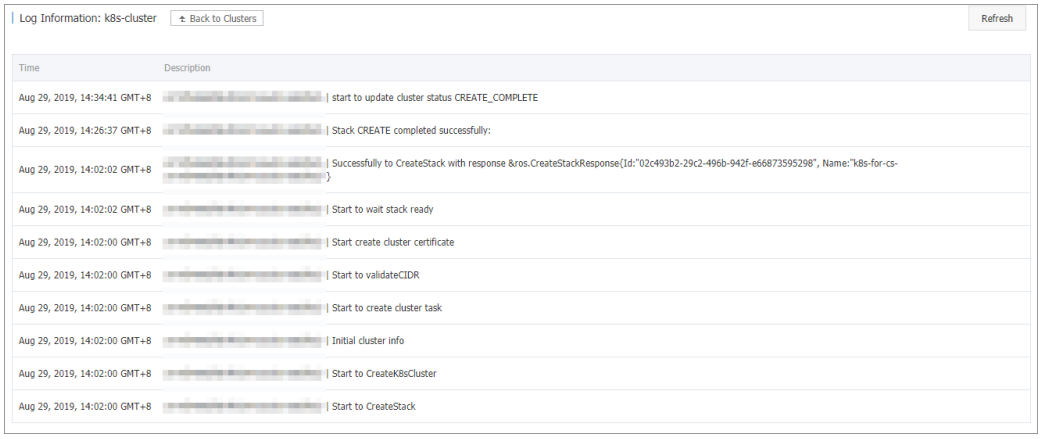
You can view the operation log of a cluster in the Container Service console.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. Find the cluster that you want to manage and click **View Logs** in the Actions column.



You can view operations performed on the cluster.



6.2.4. Connect to a cluster through kubectl

You can use the Kubernetes command line tool, kubectl, to connect to a Kubernetes cluster from a local computer.

Procedure

1. Download the latest kubectl client from the Kubernetes change log page.
2. Install and set up the kubectl client.

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

   You can use the `scp` command to securely copy the master node configuration file from the `/etc/kubernetes/kube.conf` directory of the master VM and paste it to the `$HOME/.kube/config` directory of the local computer, where the `kubectl` credentials are expected to be stored.

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

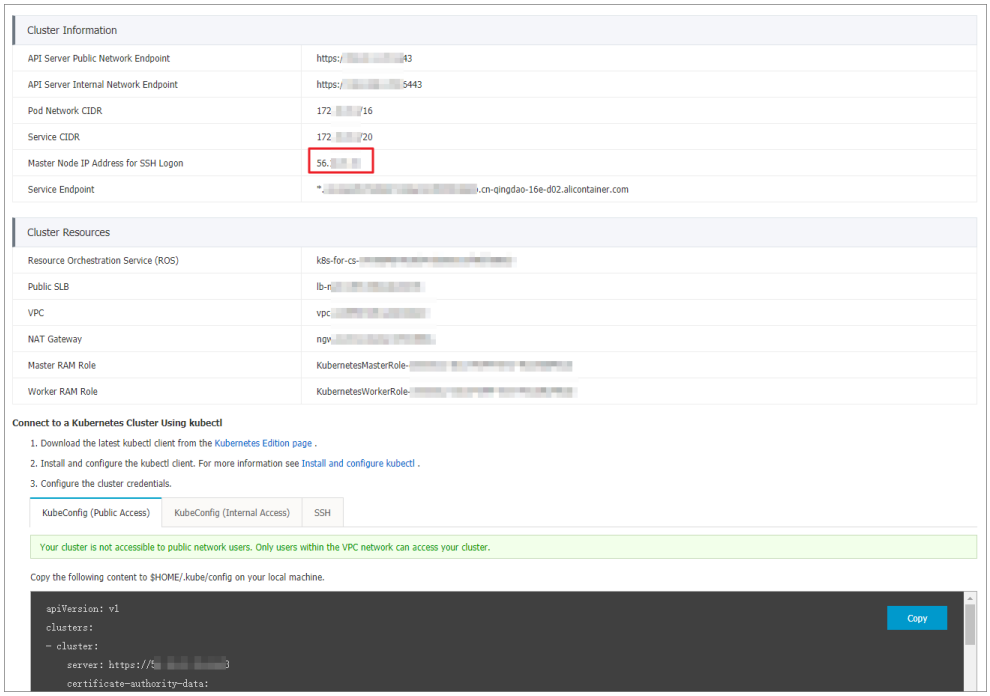
   You can find `master-public-ip` on the cluster details page.

   i. Log on to the Container Service console.

   ii. In the left-side navigation pane, click **Clusters**. The Clusters page appears.

iii. Find the target cluster and click **Manage** in the Actions column.

In the **Cluster Information** section, you can find the master node IP address.



6.2.5. Connect to a master node by using SSH

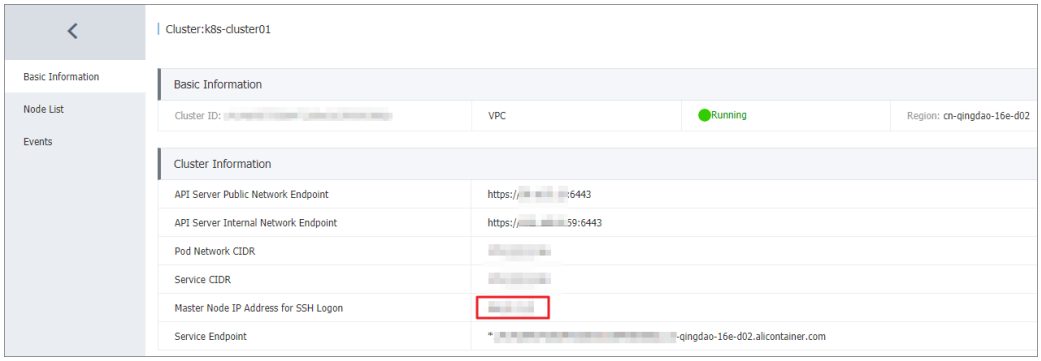
You can access a master node in a cluster by using a Secure Shell (SSH) client.

Prerequisites

- A Kubernetes cluster is created and **Use SSH to Access the Cluster from the Internet** is selected for the cluster. For more information, see Create a Kubernetes cluster.
- The SSH client can connect to the network where the cluster is deployed.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Clusters** > **Clusters** to go to the Clusters page. Find the cluster that you want to manage, and click Manage in the Actions column for the cluster.
3. The Basic Information page appears. In the Cluster Information section, you can find the IP address that is displayed in the **Master Node IP Address for SSH Logon** field.



4. Use SSH to connect to the cluster from an SSH client that has access to the cluster network.
   - If you have a leased line that connects to the cluster network over the Internet, you can use tools such as PuTTY to create an SSH connection.
   - If you have an Elastic Compute Service (ECS) instance that is connected to the Virtual Private Cloud (VPC) network of the cluster, run the following command to create an SSH connection:

```
ssh    root@ssh_ip                    #ssh_ip specifies the IP address of the maste
r node for SSH connection.
```

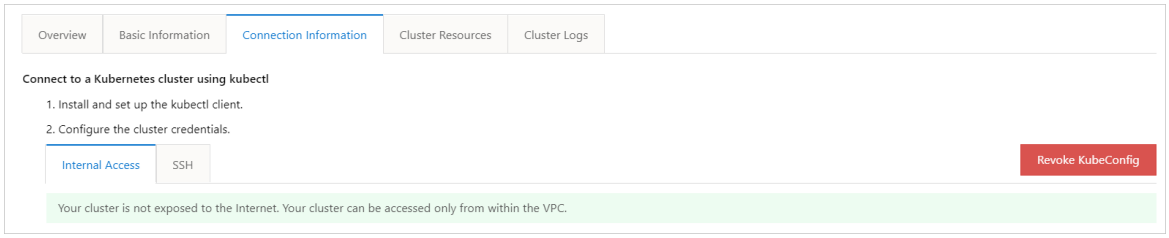
6.2.6. Revoke the kubeconfig file of a cluster

In multi-tenant scenarios, Container Service issues kubeconfig files to users that are assigned different roles. The kubeconfig files are used to connect to clusters. If an employee leaves the company or a kubeconfig file is suspected to be leaked, you can revoke the kubeconfig file to ensure cluster security. This topic describes how to revoke a kubeconfig file in the Container Service console.

Procedure

Notice After a kubeconfig file is revoked, you cannot use the kubeconfig file to connect to the cluster. Proceed with caution.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, click the name of the cluster that you want to manage, or click **Details** in the **Actions** column of the cluster.
4. On the **Cluster Information** page, click the **Connection Information** tab, and then click **Revoke KubeConfig**.



5. In the message that appears, click **OK**.

   The kubeconfig file that is used by the current account to connect to the cluster is revoked. The system then automatically assigns the account a new kubeconfig file.

6.2.7. Update the Kubernetes version of a cluster

You can go to the Clusters page to check the Kubernetes version of your cluster and check whether the current version can be updatable. This topic describes how to update the Kubernetes version of a cluster.

Considerations

- The update may fail. To ensure data security, we recommend that you create snapshots before you start the update. For more information about how to create snapshots to back up data on Elastic Compute Service (ECS) instances, see the *Create a snapshot* topic of *ECS User Guide*.
- Applications that run in the cluster are not interrupted during the update. Applications that are highly reliant on the API server may be temporarily interrupted.
- Object Storage Service (OSS) volumes that are mounted to the Kubernetes cluster by using FlexVolume 1.11.2.5 or earlier will be remounted during the update. You must recreate the pods that use OSS volumes after the update is completed.
- You can modify the configurations of the cluster during the update process. For example, you can create SWAP partitions. In this case, the update may fail.
- Do not modify the resources in the kube-upgrade namespace during the update process unless an error occurs.
- If an error occurs during the update process, the update is paused. You must troubleshoot the error and delete the failed pods in the kube-upgrade namespace. You can restart the update after the error is fixed.
- After the update is completed, we recommend that you update kubectl on your on-premises machine. Otherwise, the kubectl version may be incompatible with the API server version. As a result, the error message **invalid object doesn't have additional properties** may appear.
- You can update the Kubernetes version of a Container Service cluster only to the later version. For example, the Kubernetes version of a Container Service cluster is 1.12. If you want to update the Kubernetes version to 1.18, you must manually update the version to 1.14, 1.16, and then 1.18 in sequence.
- Container Service clusters that run Kubernetes 1.20 do not support the selfLink field. If FlexVolume is used and alicloud-nas-controller is installed in your cluster, you must update the image version of alicloud-nas-controller to v1.14.8.17-7b898e5-aliyun or later before you can update an earlier Kubernetes version to 1.20. For more information, see Container Service support for Kubernetes 1.20.
- Specific API versions are deprecated in Kubernetes 1.18. If the deprecated API versions are used in a cluster whose Kubernetes version is earlier than 1.18, you must replace these API versions with new API versions before you update the Kubernetes version to 1.18. For more information, see Container Service support for Kubernetes 1.18.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and choose **More > Upgrade Cluster**.
4. On the **Upgrade Cluster** page, click **Upgrade**.

   On the **Upgrade Cluster** page, you can view the **current Kubernetes version** and the update considerations. If the Upgrade button is not displayed on the **Upgrade Cluster** page, the current

Kubernetes version is updated to the latest.

5. In the message that appears, click **OK**.

6.2.8. Expand a cluster

This topic describes how to scale out the worker nodes of a Kubernetes cluster in the Container Service console.

Context

You cannot scale out the master nodes of a Kubernetes cluster.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to expand and choose **More > Expand** in the **Actions** column.
4. On the **Node Pools** page, select the node pool that you want to scale out and click **Scale Out** in the **Actions** column.
5. Go to the Expand page and set the required parameters.

   In this example, the number of worker nodes in the cluster is increased from three to five. The following table describes the required parameters.

Parameter	Description
Nodes to Add	Specify the number of nodes to be added to the cluster.
Region	By default, the region where the cluster is deployed is displayed.
Container Runtime	By default, the container runtime of the cluster is displayed.
VPC	By default, the virtual private cloud (VPC) of the cluster is displayed.
VSwitch	Select one or more vSwitches for the cluster. You can select at most three vSwitches that are deployed in different **zones**.
Instance Type	You can select one or more instance types. For more information, see the *Instance types* topic of *ECS User Guide*.
Selected Types	The selected instance types.
System Disk	Standard SSDs, enhanced SSDs (ESSDs), and ultra disks are supported.
Mount Data Disk	Standard SSDs, ESSDs, and ultra disks are supported.
Operating System	The operating system of the cluster.

Parameter	Description
Password	○ **Password**: Enter the password that is used to log on to the nodes. ○ **Confirm Password**: Enter the password again.
ECS Label	You can add labels to the ECS instances.
Node Label	Add labels to nodes.
Taints	Add taints to the worker nodes in the cluster.
Custom Image	You can select a custom image. After you select a custom image, all nodes in the cluster are deployed by using this image.
RDS Whitelist	Set the Relational Database Service (RDS) whitelist. Add the IP addresses of the nodes in the cluster to the RDS whitelist.
User Data	Customize the startup behaviors of ECS instances and import data to the ECS instances. The user data can be used in the following ways: ○ Run scripts during instance startup. ○ Import user data as normal data to an ECS instance for future reference.

6. Click **Submit**.

What's next

After the cluster is expanded, choose **Nodes > Nodes** in the left-side navigation pane. On the Nodes page, you can find that the number of worker nodes is increased from 3 to 5.

6.2.9. Renew a certificate

This topic describes how to renew a Kubernetes cluster certificate in the console.

Prerequisites

A Kubernetes cluster is created and the cluster certificate is about to expire.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. Select the cluster for which you want to renew the certificate and click **Update Certificate**. The **Update Certificate** message appears.

   > **Note** The **Update Certificate** button will be displayed two months before your cluster certificate expires.

4. Click **Update** and the **Confirm** page appears.
5. Click **OK**.

Result

- On the **Update Certificate** page, the following message appears: **The certificate has been updated**.
- On the **Clusters** page, the **Update Certificate** button disappears.

6.2.10. Delete a Kubernetes cluster

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

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to delete and choose **More > Delete** in the **Actions** column.
4. In the **Delete Cluster** dialog box that appears, select the resources that you want to retain, select **I understand the above information and want to delete the specified cluster**, and then click **OK**.

What's next

Resource Orchestration Service (ROS) does not have permissions to delete resources that are manually added to resource created by ROS. For example, if you manually add a vSwitch to a virtual private cloud (VPC) created by ROS, ROS cannot delete the VPC and therefore the cluster cannot be deleted.

Container Service allows you to forcibly delete clusters. If your first attempt to delete a cluster fails, you can forcibly delete the cluster and ROS resource stack. However, you still need to manually release the resources that are manually added.

An error message appears when an attempt to delete a cluster fails.

Select the cluster that you failed to delete and choose **More > Delete** in the Actions column. In the dialog box that appears, you can view the resources that are manually added. Select the **Force Delete** check box and click **OK** to delete the cluster and ROS resource stack.

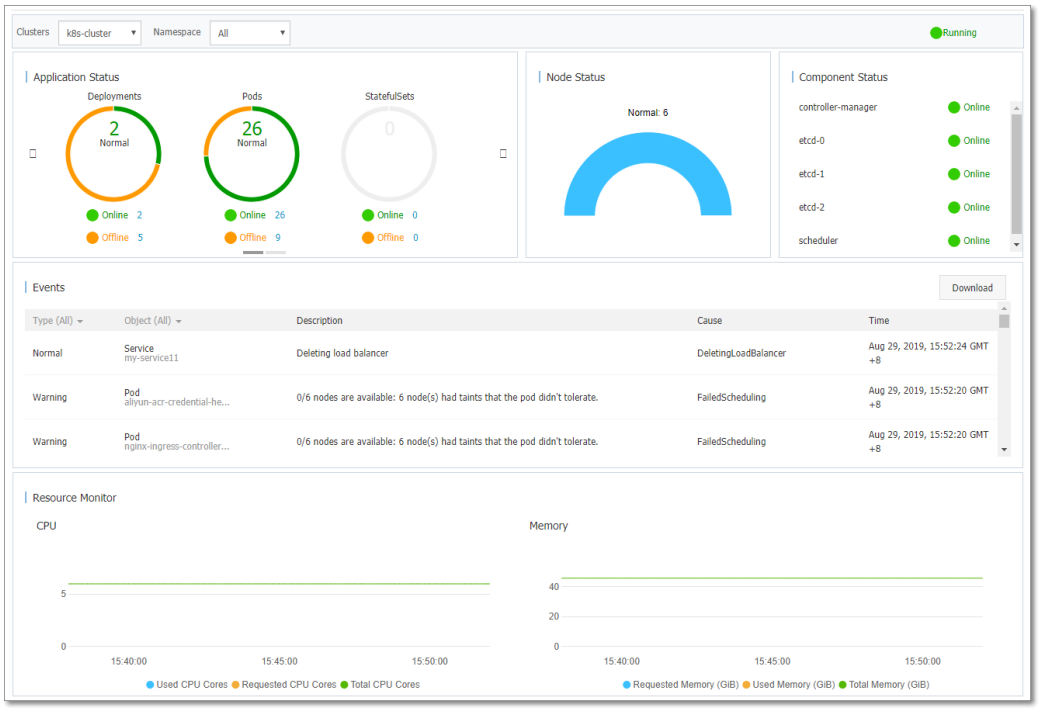
6.2.11. View cluster overview

The Container Service console provides a cluster overview page. This page displays the information such as application status, component status, and resource monitoring status. This allows you to check the health status of your cluster at your convenience.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Overview**. The Overview page appears.
3. Select the target cluster and namespace. You can view the application status, component status, and resource monitoring charts.
   - **Application Status**: displays the statuses of the deployments, pods, and replica sets that are running in the cluster. Green sections indicate a normal state and yellow sections indicate an exception state.

- **Node Status**: displays the statuses of the nodes in the cluster.
- **Component Status**: Components are deployed in the kube-system namespace. Core components are used, such as the scheduler, controller-manager, and etcd.
- **Events**: displays events such as warnings and errors. If no events are displayed, the cluster is running in the normal state.
- **Monitoring**: displays CPU and memory monitoring charts. CPU usage is measured in cores or millicores and accurate to three decimal places. A millicore is one thousandth of a core. Memory usage is measured in GiB and accurate to three decimal places. For more information, see Meaning of CPU and Meaning of memory.



6.3. Nodes

6.3.1. Add existing ECS instances to a Container Service cluster

You can add existing Elastic Compute Service (ECS) instances to a Container Service cluster in the Container Service console. ECS instances can be added only as worker nodes to a Container Service cluster. This topic describes how to manually add ECS instances to a Container Service cluster. This topic also describes how to configure Container Service to automatically add ECS instances to a Container Service cluster.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- The ECS instances that you want to add are added to the security group of the nodes in your cluster. The security group is automatically created when the cluster is initialized. For more information, see the *Add an ECS instance to a security group* topic in the *Security groups* chapter of *ECS User Guide*.

Limits

- Make sure that you have a sufficient node quota in the cluster.
- The ECS instances that you want to add to the Container Service cluster must be deployed in the same region and virtual private cloud (VPC) as the cluster.
- The ECS instances that you want to add must belong to the account that owns the cluster.
- Nodes that run the following operating systems can be added to a Container Service cluster:
  - Alibaba Cloud Linux 2
  - CentOS 7.x

> **Note** CentOS 8.x or later are not supported.

Limits on ECS instances

Number of security groups

When an existing instance is added to a node pool, the instance is also added to the security group of the node pool. The system skips this step if the instance is already in the security group. Make sure that the number of security groups to which the instance belongs does not exceed the upper limit after the instance is added to the node pool. For more information about the limits of ECS instances, see the *Limits* topic of *ECS User Guide*.

Limits on security group rules

A security group acts as a virtual firewall to control the inbound and outbound traffic of ECS instances to improve security.

Security groups are classified into basic security groups, advanced security groups, and managed security groups. Managed security groups are created by cloud services. The rules of a managed security group can be modified only by the cloud service that created the managed security group. When you create a node pool, do not select a managed security group. Otherwise, you fail to create the node pool. The following table compares basic security groups and advanced security groups.

Feature	Basic security group	Advanced security group
Access control policy when the security group does not contain rules	• Inbound: denies all requests. • Outbound: allows all requests.	• Inbound: denies all requests. • Outbound: denies all requests.
Maximum number of private IP addresses	2000	65536
Mutual access between instances in the same security group	By default, instances in the same security group can access each other over the internal network.	By default, instances in the same security group are isolated from each other by internal networks. You must manually add security group rules to allow mutual access between the instances.
Access to or from other security groups	Access to or from other security groups is allowed.	Rules cannot be added to control access to or from other security groups.

When you create a node pool, the system automatically adds rules to the security group of the node pool to enable communication within the cluster. For more information about how to configure security groups, see the *Add security group rules* topic in the *Security groups* chapter of *ECS User Guide*.

When you add an instance to a security group, make sure that the following rules do not conflict with the rules of the security groups to which the instance already belongs. Otherwise, pods cannot communicate with each other.

- The pod CIDR block
- The secondary IPv4 CIDR block of the VPC

Automatically add ECS instances

In auto mode, all ECS instances that are available within your account are listed. You can select, configure, and add one or more ECS instances to a cluster in the Container Service console. After you complete the configurations, the ECS instances are automatically added to the cluster.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Nodes > Node Pools**.
5. On the **Node Pools** page, find the node pool to which you want to add nodes and click **Add Existing Node** in the **Actions** column.
6. On the **Add Existing ECS Instance** wizard page, select the auto mode to automatically add ECS instances to the cluster.

   Set **Mode** to **Auto** and select the ECS instances that you want to add in the Select Existing ECS Instance section.
7. Click **Next Step** and set the parameters on the **Specify Instance Information** wizard page.

Parameter	Description
Cluster ID/Name	Information about the cluster to which the instances are to be added. This parameter is automatically set.
Data Disk	Specify whether to store the container and image data on a data disk. ◦ If the ECS instances have data disks mounted and the file system of the last data disk is not initialized, the system automatically formats the data disk to ext4. Then, the system uses the disk to store the data in */var/lib/docker* and */var/lib/kubelet*. **Note** After the data disk is formatted, the data that is stored on the disk is deleted. Make sure that you have backed up the data before you add the ECS instances to the Container Service cluster. ◦ If no data disk is attached to the ECS instances, the system does not purchase a new data disk.
Password	Enter a **password** and then **enter the password again**.

Parameter	Description
Retain Instance Name	By default, **Retain Instance Name** is turned on. If you do not want to retain the instance name, you can turn off **Retain Instance Name**. After you disable this feature, the nodes are renamed based on the node naming rules.
Instance Information	The IDs and names of the instances to be added.

8. Click **Next Step**. In the **Confirm** message, click **Confirm**.

Manually add ECS instances

> **Notice** ECS instances that are manually added to a Container Service cluster are not released when the Container Service cluster is deleted.

In manual mode, you must obtain the installation command, log on to an ECS instance, and then run the command to add the ECS instance to a Container Service cluster. You can add only one ECS instance at a time.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Nodes > Node Pools**.
5. On the **Node Pools** page, find the node pool to which you want to add nodes and click **Add Existing Node** in the **Actions** column.
6. On the **Select Existing ECS Instance** wizard page, select the manual mode to add ECS instances to the cluster.

   Set **Mode** to **Manual** and select the ECS instances that you want to add in the Select Existing ECS Instance section.
7. Click **Next Step** and set the parameters on the **Specify Instance Information** wizard page.

Parameter	Description
Cluster ID/Name	Information about the cluster to which the instances are to be added. This parameter is automatically set.

Parameter	Description
Data Disk	Specify whether to store the container and image data on a data disk. ○ If the ECS instances have data disks mounted and the file system of the last data disk is not initialized, the system automatically formats the data disk to ext4. Then, the system uses the disk to store the data in */var/lib/docker* and */var/lib/kubelet*. **Note** After the data disk is formatted, the data that is stored on the disk is deleted. Make sure that you have backed up the data before you add the ECS instances to the Container Service cluster. ○ If no data disk is attached to the ECS instances, the system does not purchase a new data disk.
Password	Enter a **password** and then **enter the password again**.
Retain Instance Name	By default, **Retain Instance Name** is turned on. If you do not want to retain the instance name, you can turn off **Retain Instance Name**. After you disable this feature, the nodes are renamed based on the node naming rules.
Instance Information	The IDs and names of the instances to be added.

8. Click **Next Step** to go to the **Complete** wizard page. On the **Complete** wizard page, copy the command and click **Done**.
9. Log on to the ECS console. For more information, see the *Log on to the ECS console* topic in the *Getting Started* chapter of *ECS User Guide*.
10. In the left-side navigation pane, choose **Instances & Images > Instances**. Select a region and then select the ECS instance that you want to add.
11. Click **Connect** in the Actions column to connect to the ECS instance by using Virtual Network Computing (VNC). For more information, see the *Connect to an instance by using VNC* topic in the *Getting Started* chapter of *ECS User Guide*.
12. On the connection page, follow the instructions and paste the command that you copied in . Then, click **Run** to execute the script.
    After the script is executed, the ECS instance is added to the cluster.

Result

1. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
2. On the **Node Pools** page, find the node pool that you want to manage and click **Details** in the **Actions** column.
3. On the node pool details page, click the **Nodes** tab.

   You can view the newly added nodes.

6.3.2. View nodes

You can view nodes of a Kubernetes cluster by using the Container Service console, kubectl, or Kubernetes Dashboard.

View nodes by using kubectl

> **Note** To view the nodes in a cluster by using kubectl, you must Connect to a Kubernetes cluster through kubectl.

Connect to a cluster by using kubectl and run the following command to view the nodes in the cluster:

```
kubectl get nodes
```

Sample output:

```
$ kubectl get nodes
    NAME                       STATUS    AGE       VERSION
    iz2ze2n6ep53tch701yh9zz    Ready     19m       v1.6.1-2+ed9e3d33a07093
    iz2zeafr762wibijx39e5az    Ready     7m        v1.6.1-2+ed9e3d33a07093
    iz2zeafr762wibijx39e5bz    Ready     7m        v1.6.1-2+ed9e3d33a07093
    iz2zef4dnn9nos8elyr32kz    Ready     14m       v1.6.1-2+ed9e3d33a07093
    iz2zeitvvo8enoreufstkmz    Ready     11m       v1.6.1-2+ed9e3d33a07093
```

View nodes by using the Container Service console

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes** to view the nodes in the cluster.

6.3.3. Manage node labels

You can manage node labels in the Container Service console. You can add a label to multiple nodes at a time, filter nodes by label, and delete labels.

You can use labels to schedule nodes. For more information, see Set node scheduling.

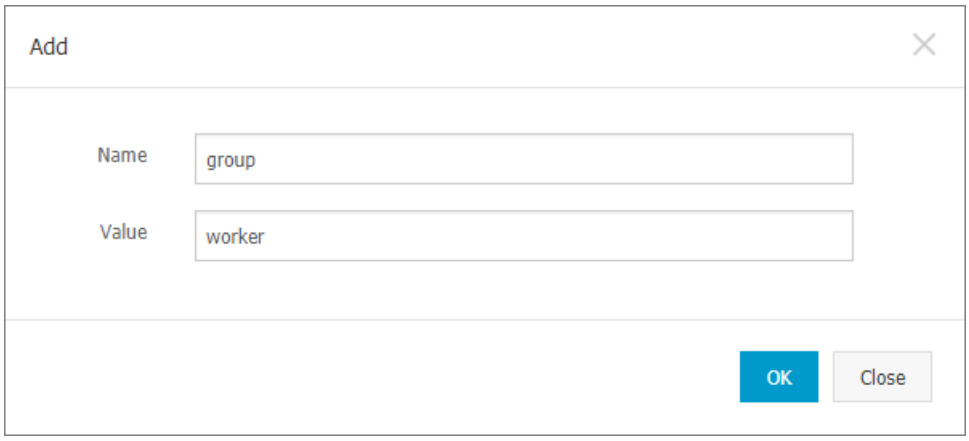
Prerequisites

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

Add a label to multiple nodes at a time

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner.

6. Select multiple nodes and click **Add Label**.
7. In the dialog box that appears, enter the name and value of the label and click **OK**.



On the Labels tab, you can find that the selected nodes have the same label.

Delete a label

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner.
6. Select a node, find the label that you want to delete, and then click the icon. In the message that appears, click **Confirm**.

   After the label is deleted, it is removed from the Labels column.

6.3.4. Set node schedulability

You can mark a node as schedulable or unschedulable in the Container Service console. This allows you to optimize the distribution of the loads on each node. This topic describes how to set node schedulability.

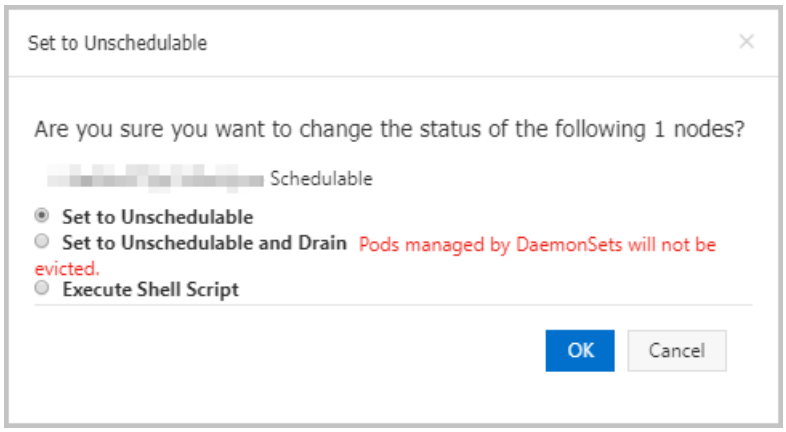
Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the Nodes page, select the node that you want to manage and click **Drain/Set to Unschedulable** in the Actions column.
6. In the dialog box that appears, you can change the status of the node.
   - If you select **Set to Unschedulable**, pods will not be scheduled to this node when you deploy

new applications.

- If you select **Set to Unschedulable and Drain**, pods will not be scheduled to this node when you deploy new applications. Pods on this node will be evicted, except for the pods that are managed by DaemonSets.

  In this example, **Set to Unschedulable** is selected.



- Click **OK**.

  The status of the node is changed to Unschedulable.

What's next

Pods will not be scheduled to the node when you deploy new applications.

6.3.5. Remove a node

To restart or release an Elastic Compute Service (ECS) node in a cluster, you must first remove the node from the cluster. This topic describes how to remove a node.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- You can connect to the Kubernetes cluster by using kubectl. For more information, see Connect to a Kubernetes cluster through kubectl.

Context

- When you remove a node, pods that run on the node are migrated to other nodes. This may cause service interruption. We recommend that you remove nodes during off-peak hours.
- Unknown errors may occur when you remove nodes. Before you remove nodes, we recommend that you back up data on these nodes.
- Nodes remain in the unschedulable state when they are being removed.
- You can remove only worker nodes. You cannot remove master nodes.

Procedure

1. Run the following command to migrate the pods on the node that you want to remove to other nodes.

   > **Note** Make sure that the other nodes have sufficient resources for these pods.

```
kubectl drain node-name
```

Note *node-name* must be in the format of *your-region-name.node-id*.

- *your-region-name* specifies the region where the cluster that you want to manage is deployed.
- *node-id* specifies the ID of the ECS instance where the node to be removed is deployed. Example: *cn-hanghzou.i-xxx*.

2. Log on to the Container Service console.
3. In the left-side navigation pane, click **Clusters**.
4. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
5. In the left-side navigation pane of the details page, choose **Nodes > Node**.
6. Find the node that you want to remove and choose **More > Remove** in the **Actions** column.

Note To remove multiple nodes at a time, select the nodes that you want to remove on the **Nodes** page and click **Batch Remove**.

7. (Optional)Set parameters in the Remove Node dialog box.
   - Select **Drain the Node** to migrate the pods on the node to other nodes. If you select this option, make sure that the other nodes have sufficient resources for these pods.
   - Select **Release ECS Instance** to release the ECS instance where the node is deployed.

Note
- Select this option to release only pay-as-you-go ECS instances.
- Subscription ECS instances are automatically released after the subscription expires.
- If you do not select **Release ECS Instance**, you are still billed for the ECS instance where the node is deployed.

8. In the **Remove Node** message, click **OK**.

6.3.6. View node resource usage

You can view the resource usage of the nodes in a cluster in the Container Service console.

Prerequisites

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

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.

4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the Nodes page, find the node that you want to manage and choose **More > Details** in the **Actions** column.

   You can view the request rate and usage rate of CPU and memory resources on each node.

   - CPU request rate = sum (The amount of CPU resources requested by all pods on the node) / Total CPU resources of the node
   - CPU usage rate = sum (The amount of CPU resources used by all pods on the node) / Total CPU resources of the node
   - Memory request rate = (The amount of memory resources requested by all pods on the node) / Total memory resources of the node
   - Memory usage rate = sum (The amount of memory resources used by all pods on the node) / Total memory resources of the node

   > **Note**
   > - You can adjust the workload of a node based on the resource usage. For more information, see Set node scheduling.
   > - When the request or usage rate of a node reaches 100%, pods are not scheduled to the node.

6.3.7. Node pools

6.3.7.1. Create a node pool

Container Service allows you to create node pools and manage nodes in a cluster by node pool. For example, you can centrally manage the labels and taints of the nodes in a node pool. This topic describes how to create a node pool in the Container Service console.

Prerequisites

- A Container Service cluster is created. For more information, see Create a Kubernetes cluster.
- The Kubernetes version of the cluster must be later than 1.9.

> **Note** By default, you can deploy at most 100 nodes in a Container Service cluster.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Nodes > Node Pools**.
5. In the upper-right corner of the **Node Pools** page, click **Create Node Pool**.
6. On the **Create Node Pool** page, click **Show Advanced Options** and configure advanced settings.

For more information about the parameters, see Create a Kubernetes cluster. The following list describes some of the parameters:

- Name: the name of the node pool.
- Quantity: Specify the number of nodes that you want to add to the node pool. If you do not want to add nodes to the node pool, set this parameter to 0.
- Public IP: If you select Assign a Public IPv4 Address to Each Node, public IPv4 addresses are assigned to the nodes in the node pool. You can connect to the nodes by using the assigned IP addresses.
- ECS Label: Add labels to the ECS instances.
- Node Label: Add labels to the nodes in the node pool.
- CPU Policy: Set the CPU policy.
  - none: This policy indicates that the default CPU affinity is used. This is the default policy.
  - static: This policy allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
- Custom Security Group: Click **Select a security group** to select a custom security group. For more information about security groups, see the *Create a security* topic of *ECS User Guide*.
- Custom Node Name: Specify whether to use custom node names.

  A custom node name consists of a prefix, an IP substring, and a suffix.
  - The prefix and suffix can contain multiple parts that are separated by periods (.). Each part can contain lowercase letters, digits, and hyphens (-), and must start and end with a lowercase letter or digit.
  - The IP substring length specifies the number of digits to be truncated from the end of the node IP address. The IP substring length ranges from 5 to 12.

  For example, if the node IP address is 192.1xx.x.xx, the prefix is aliyun.com, the IP substring length is 5, and the suffix is test, the node name will be aliyun.com00055test.

7. Click **Confirm Order**.
   On the **Node Pools** page, check the **status** of the node pool. If the node pool is in the **Initializing** state, the node pool is being created. After the node pool is created, the node pool is in the **Active state**.

What's next

After the node pool is created, go to the **Node Pools** page, find the cluster, and then click **Details** in the **Actions** column to view the details of the node pool.

6.3.7.2. Scale out a node pool

You can use a node pool to manage multiple nodes in a Kubernetes cluster as a group. For example, you can centrally manage the labels and taints of the nodes in a node pool. This topic describes how to scale out a node pool in the Container Service console.

Prerequisites

A node pool is created. For more information, see Create a node pool.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. On the **Node Pools** page, select the node pool that you want to manage and click **Scale Out** in the **Actions** column.
6. In the dialog box that appears, set the number of nodes that you want to add to the node pool.
7. (Optional)In the dialog box that appears, click **Modify Node Pool Settings** to modify the configurations of the node pool.

   For more information, see Create a Kubernetes cluster. The following list describes some of the parameters:

   - Public IP: If you select Assign a Public IPv4 Address to Each Node, public IPv4 addresses are assigned to the nodes in the node pool. You can connect to the nodes by using the assigned IP addresses.
   - ECS Label: Add labels to the ECS instances.
   - Node Label: Add labels to the nodes in the node pool.
   - Taints: Add taints to all worker nodes in the Kubernetes cluster.

   > **Note**
   > - If you select **Synchronize Node Labels and Taints**, the added labels and taints are synchronized to both existing and newly added nodes.
   > - If you select **Set New Nodes to Unschedulable**, the nodes are unschedulable when they are added to the cluster.

8. Click **Submit**.
   On the **Node Pools** page, the **state** of the node pool is **Scaling**. This indicates that the scale-out event is in progress. After the scale-out event is completed, the **state** of the node pool changes to **Active**.

What's next

Click **Details** in the Actions column for the node pool. On the **Nodes** tab, you can check the nodes that are added to the node pool.

6.3.7.3. Schedule an application pod to a specific node pool

Label is an important concept of Kubernetes. Services, Deployments, and pods are associated with each other by labels. You can configure pod scheduling policies based on node labels. This allows you to schedule pods to nodes that have specific labels. This topic describes how to schedule an application pod to a specific node pool.

Procedure

1. Add a label to the nodes in a node pool.

Container Service allows you to manage a group of cluster nodes by using a node pool. For example, you can centrally manage the labels and taints of the nodes in a node pool. For more information about how to create a node pool, see Create a node pool.

i. Log on to the Container Service console.
ii. In the left-side navigation pane, click **Clusters**.
iii. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
iv. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
v. In the upper-right corner of the **Node Pools** page, click **Create Node Pool**.
vi. In the Create Node Pool dialog box, click **Show Advanced Options** and click  on the right of **Node Label** to add labels to nodes.

In this example, the pod: nginx label is added.

You can also click **Scale Out** on the right side of a node pool to update or add labels for the nodes. If automatic scaling is enabled for a node pool, click **Modify** on the right side of the node pool to update or add labels for the nodes.

2. Configure a scheduling policy for an application pod.

After the preceding step is completed, the pod: nginx label is added to the nodes in the node pool. You can set the nodeSelector or nodeAffinity field in pod configurations to ensure that an application pod is scheduled to nodes with matching labels in a node pool. Perform the following steps:

- Set nodeSelector.

  nodeSelector is a field in the spec section of pod configurations. Add the pod: nginx label to nodeSelector. Sample template:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment-basic
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      nodeSelector:
        pod: nginx      # After you add the label in this field, the application pod
can run only on nodes with this label in the node pool.
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

- Set nodeAffinity.

  You can also use nodeAffinity to schedule an application pod based on your requirements. nodeAffinity supports the following scheduling policies:

  ```
  - requiredDuringSchedulingIgnoredDuringExecution
  ```

  If this policy is used, a pod can be scheduled only to a node that meets the match rules. If no node meets the match rules, the system retries until a node that meets the rules is found. IgnoreDuringExecution indicates that if the label of the node where the pod is deployed changes and no longer meets the match rules, the pod continues to run on the node.

  ```
  - requiredDuringSchedulingRequiredDuringExecution
  ```

  If this policy is used, the pod can be scheduled only to a node that meets the match rules. If no node meets the rules, the system retries until a node that meets the rules is found. RequiredDuringExecution indicates that if the label of the node where the pod is deployed changes and no longer meets the match rules, the system redeploys the pod to another node that meets the rules.

  ```
  - preferredDuringSchedulingIgnoredDuringExecution
  ```

  If this policy is used, the pod is preferably scheduled to a node that meets the match rules. If no node meets the rules, the system ignores the rules.

```
- preferredDuringSchedulingRequiredDuringExecution
```

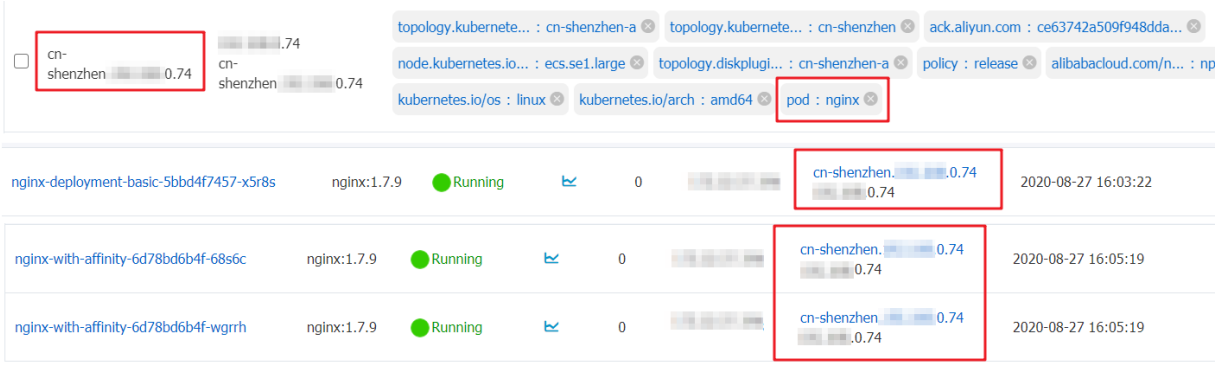
If this policy is used, the pod is preferably scheduled to a node that meets the match rules. If no node meets the rules, the system ignores the rules. RequiredDuringExecution indicates that if the label of a node where the pod is deployed changes and still meets the match rules, the system reschedules the pod to a node that meets the match rules.

In the following example, the requiredDuringSchedulingIgnoredDuringExecution policy is used to ensure that the application pod always runs on a node in a specific node pool.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-with-affinity
  labels:
    app: nginx-with-affinity
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-with-affinity
  template:
    metadata:
      labels:
        app: nginx-with-affinity
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: pod
                operator: In      # This policy ensures that the application pod can
run only on a node that has the pod: nginx label.
                values:
                - nginx
      containers:
      - name: nginx-with-affinity
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Result

In the preceding example, all application pods are scheduled to the xxx.xxx.0.74 node. This node has the pod: nignx label.



6.4. Storage

6.4.1. Overview

You can create disk volumes and Object Storage Service (OSS) volumes in the Container Service console to enable data persistence for stateful applications.

Both statically provisioned volumes and dynamically provisioned volumes are supported. The following table describes the types of volumes that different storage services support.

Alibaba Cloud storage service	Statically provisioned volume	Dynamically provisioned volume
Alibaba Cloud disks	You can use statically provisioned disk volumes in the following ways: • Directly mount as volumes • By creating a persistent volume (PV) and a persistent volume claim (PVC)	Supported

Apsara File Storage NAS (NAS)	You can use statically provisioned NAS volumes in the following ways:
	• Use the FlexVolume plug-in
	  ◦ Directly mount as volumes
	  ◦ By creating a PV and a PVC
	• Use the Network File System (NFS) driver of Kubernetes
	Note You must install an NFS client on the node.
		Supported
		Note You must install an NFS client on the node.

OSS	You can use statically provisioned OSS volumes in the following ways: • Directly mount as volumes • By creating a PV and a PVC	Not supported

6.4.2. Mount disk volumes

You can mount disks as volumes.

Container Service allows you to mount disks as persistent volumes (PVs) in Kubernetes clusters.

Disks can be mounted to Kubernetes clusters as the following volume types:

- Statically provisioned disk volumes

  You can use statically provisioned disk volumes in the following ways:

  - Mount disks as volumes
  - Mount disk volumes by creating a PV and a persistent volume claim (PVC)

- Dynamically provisioned disk volumes

Usage notes

- You can mount a disk only to one pod.
- Before you mount a disk to a pod, you must create the disk and obtain its disk ID.

  The disk must meet the following capacity requirements:

  - If the disk is a basic disk, it must be at least 5 GiB in size.
  - If the disk is an ultra disk, it must be at least 20 GiB in size.
  - If the disk is a standard SSD, it must be at least 20 GiB in size.

- volumeId: the ID of the disk that you want to mount. The value must be the same as those of volumeName and PV Name.
- A disk can be mounted only to a node that is deployed in the same zone as the disk.
- Only pay-as-you-go disks can be mounted. If you change the billing method of an Elastic Compute Service (ECS) instance in the cluster from pay-as-you-go to subscription, you cannot change the billing method of its disks to subscription. Otherwise, the disks cannot be mounted to the cluster.

Statically provisioned disk volumes

You can mount disks as volumes or by creating PVs and PVCs.

Prerequisites

A disk is created in the ECS console.

- **Mount a disk as a volume**

  Use the following *disk-deploy.yaml* file to create a pod:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-disk-deploy
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx-flexvolume-disk
        image: nginx
        volumeMounts:
          - name: "d-bp1j17ifxfasvts3tf40"
            mountPath: "/data"
      volumes:
        - name: "d-bp1j17ifxfasvts3tf40"
          flexVolume:
            driver: "alicloud/disk"
            fsType: "ext4"
            options:
              volumeId: "d-bp1j17ifxfasvts3tf40"
```

- **Mount disk volumes by creating a PV and a PVC**

  i. **Create a PV of the disk type**

  You can create a PV of the disk type in the Container Service console or by using a YAML file.

- **Create a PV by using a YAML file**

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

  > **Note** The PV name must be the same as the disk ID.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: d-bp1j17ifxfasvts3tf40
  labels:
    failure-domain.beta.kubernetes.io/zone: cn-hangzhou-b
    failure-domain.beta.kubernetes.io/region: cn-hangzhou
spec:
  capacity:
    storage: 20Gi
  storageClassName: disk
  accessModes:
    - ReadWriteOnce
  flexVolume:
    driver: "alicloud/disk"
    fsType: "ext4"
    options:
      volumeId: "d-bp1j17ifxfasvts3tf40"
```

- **Create a PV in the console**
  a. Log on to the Container Service console.
  b. In the left-side navigation pane, click **Clusters**.
  c. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
  d. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.
  e. On the **Persistent Volumes** page, click **Create** in the upper-right corner.
  f. In the Create PV dialog box, set the parameters.

PV parameters

Parameter	Description
PV Type	In this example, **Cloud Disk** is selected.
Volume Plug-in	Displays the supported storage drivers.
Access Mode	By default, ReadWriteOnce is selected.
Disk ID	Select a disk that is in the same region and zone as your cluster.
File System Type	Select the file system of the disk. Supported file systems are ext4, ext3, xfs, and vfat. Default value: ext4.
Label	Add labels to the PV.

  g. Click **Create**.

ii. **Create a PVC**

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

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

iii. **Create a pod**

Use the following *disk-pod.yaml* file to create a pod:

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

Dynamically provisioned disk volumes

To mount a disk as a dynamically provisioned volume, you must create a StorageClass of the disk type and specify the StorageClass in the storageClassName field of the PVC.

1. **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
   ```

   Required parameters:

   - provisioner: Set this parameter to alicloud/disk. This indicates that the Provisioner plug-in is used to create the StorageClass.
   - type: Specify the disk type. Valid values: cloud, cloud_efficiency, cloud_ssd, and available. If you set this parameter to available, the system attempts to create a disk in the following order: ultra disk, standard SSD, and basic disk. The system stops trying until a disk is created.
   - regionid: Specify the region of the disk.
   - zoneid: Specify the zone of the disk.

2. **Create a PVC**

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: disk-common
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: 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
    volumeMounts:
      - name: disk-pvc
        mountPath: "/mnt"
  restartPolicy: "Never"
  volumes:
    - name: disk-pvc
      persistentVolumeClaim:
        claimName: disk-common
```

Default options

By default, Kubernetes clusters support the following types of StorageClass:

- alicloud-disk-common: basic disk.
- alicloud-disk-efficiency: ultra disk.
- alicloud-disk-ssd: standard SSD.
- alicloud-disk-available: This option ensures high availability. The system attempts to create an ultra disk first. If no ultra disk is available in the specified zone, the system attempts to create a standard SSD. If no standard SSD is available, the system attempts to create a basic disk.

3. **Create a multi-instance StatefulSet by using a disk**

   We recommend that you use the volumeClaimTemplates parameter. This parameter allows the system to dynamically create PVCs and PVs. PVCs are associated with corresponding PVs.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    name: web
  clusterIP: None
  selector:
    app: nginx
---
apiVersion: apps/v1beta2
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: disk-common
          mountPath: /data
  volumeClaimTemplates:
  - metadata:
      name: disk-common
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "alicloud-disk-common"
      resources:
        requests:
          storage: 10Gi
```

6.4.3. Mount NAS volumes

Container Service allows you to mount Apsara File Storage NAS (NAS) file systems as persistent volumes (PVs) in Container Service clusters.

NAS file systems can be mounted to Kubernetes clusters as the following volume types:

- Statically provisioned NAS volumes

  You can statically provision NAS volumes in the following ways:

  - Use the FlexVolume plug-in
    - Directly mount NAS file systems as volumes
    - Mount NAS file systems by creating a PV and a persistent volume claim (PVC)
  - Use the Network File System (NFS) driver

- Dynamically provisioned NAS volumes

Prerequisites

A NAS file system is created in the NAS console and a mount target is added. The mount target is used to mount the NAS file system to the Kubernetes cluster. The NAS file system and your cluster are deployed in the same virtual private cloud (VPC).

Statically provisioned NAS volumes

You can use the FlexVolume plug-in provided by Alibaba Cloud or the NFS driver provided by Kubernetes to mount NAS file systems.

Use the FlexVolume plug-in

You can use the FlexVolume plug-in to directly mount a NAS file system as a volume. You can also mount a NAS file system by creating a PV and a PVC.

> **Note**
>
> - NAS is a shared storage service. You can mount a NAS file system to multiple pods.
> - server: the mount target of the NAS volume.
> - path: the directory to which the NAS volume is mount. You can specify a subdirectory. If the specified subdirectory does not exist, the system automatically creates the subdirectory.
> - vers: the version of the NFS protocol. Version 4.0 is supported.
> - mode: the access permissions on the directory of the NAS file system. If the root directory of the NAS file system is specified, you cannot modify the access permissions. If the NAS file system stores a large amount of data, the mounting operation may be time-consuming or even fail. Therefore, we recommend that you do not set the mode parameter.

Mount a NAS file system as a volume

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

```
apiVersion: v1
kind: Pod
metadata:
  name: "flexvolume-nas-example"
spec:
  containers:
    - name: "nginx"
      image: "nginx"
      volumeMounts:
        - name: "nas1"
          mountPath: "/data"
  volumes:
    - name: "nas1"
      flexVolume:
        driver: "alicloud/nas"
        options:
          server: "0cd8b4a576-grs79.cn-hangzhou.nas.aliyuncs.com"
          path: "/k8s"
          vers: "4.0"
```

Mount a NAS file system by creating a PV and a PVC

Step 1: Create a PV

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

- Create a PV by using a YAML file.

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

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

- Create a PV in the console

  i. Log on to the Container Service console.

  ii. In the left-side navigation pane, click **Clusters**.

  iii. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.

  iv. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.

v. On the **Persistent Volumes** page, click **Create** in the upper-right corner.

vi. In the Create PV dialog box, set the parameters.

Parameter	Description
PV Type	Select **NAS**.
Volume Name	Enter a name for the PV. The name must be unique in the cluster.
Volume Plug-in	By default, CSI is selected.
Capacity	Specify the capacity of the PV. The capacity of the PV cannot exceed the capacity of the disk.
Access Mode	By default, ReadWriteMany is selected.
Mount Target Domain Name	The domain name of the mount target that is used to mount the NAS file system to the cluster.
Subdirectory	Enter a subdirectory in the NAS file system. The subdirectory must start with a forward slash (/). If you set this parameter, the PV is mounted to the specified subdirectory. ■ If the specified subdirectory does not exist, the system automatically creates the subdirectory in the NAS file system. ■ If you do not set this parameter, the PV is mounted to the root directory of the NAS file system.
Version	The version of the NFS protocol. Version 3 and 4.0 are supported. By default, version 3 is used. We recommend that you use version 3.
Label	Add labels to the PV.

vii. Click **Create**.

Step 2: Create a PVC

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

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

Step 3: Create a pod

Use the following *nas-pod.yaml* file to create a pod:

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

Use the NFS driver

Step 1: Create a NAS file system

Log on to the NAS console. For more information about how to log on to the NAS console, see the *Create a NAS file system* chapter of the *NAS User Guide*.

> **Note** The NAS file system that you want to create and the Kubernetes cluster must be deployed in the same region.

In this example, the following mount target is used: `055f84ad83-ixxxx.cn-hangzhou.nas.aliyuncs.com`.

Step 2: Create a PV

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

- **Create a PV by using a YAML template**

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

  Run the following command to create a PV that uses the NAS file system:

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

- **Create a PV by using the console**

  For more information, see Mount a NAS file system by creating a PV and a PVC.

Step 3: Create a PVC

Create a PVC and associate the PVC with the PV that is created in Step 2.

```
root@master # cat << EOF | kubectl apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
     name: nasclaim
spec:
     accessModes:
       - ReadWriteMany
     resources:
       requests:
         storage: 8Gi
EOF
```

Step 4: Create a pod

Create an application to use the PV.

```
root@master # cat << EOF |kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
     name: mypod
spec:
     containers:
       - name: myfrontend
         image: registry.aliyuncs.com/spacexnice/netdia:latest
         volumeMounts:
         - mountPath: "/var/www/html"
           name: mypd
     volumes:
       - name: mypd
         persistentVolumeClaim:
           claimName: nasclaim
EOF
```

The NAS file system is mounted to the application that runs in the pod.

Dynamically provisioned NAS volumes

If you want to dynamically provision NAS volumes, you must install a driver plug-in and configure a NAS mount target.

Install the plug-in

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: alicloud-nas
provisioner: alicloud/nas
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: alicloud-nas-controller
  namespace: kube-system
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: run-alicloud-nas-controller
subjects:
  - kind: ServiceAccount
    name: alicloud-nas-controller
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: alicloud-disk-controller-runner
  apiGroup: rbac.authorization.k8s.io
---
kind: Deployment
apiVersion: extensions/v1beta1
```

```
metadata:
  name: alicloud-nas-controller
  namespace: kube-system
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: alicloud-nas-controller
    spec:
      tolerations:
      - effect: NoSchedule
        operator: Exists
        key: node-role.kubernetes.io/master
      - effect: NoSchedule
        operator: Exists
        key: node.cloudprovider.kubernetes.io/uninitialized
      nodeSelector:
         node-role.kubernetes.io/master: ""
      serviceAccount: alicloud-nas-controller
      containers:
        - name: alicloud-nas-controller
          image: registry.cn-hangzhou.aliyuncs.com/acs/alicloud-nas-controller:v1.8.4
          volumeMounts:
          - mountPath: /persistentvolumes
            name: nfs-client-root
          env:
            - name: PROVISIONER_NAME
              value: alicloud/nas
            - name: NFS_SERVER
              value: 0cd8b4a576-mmi32.cn-hangzhou.nas.aliyuncs.com
            - name: NFS_PATH
              value: /
      volumes:
        - name: nfs-client-root
          nfs:
            server: 0cd8b4a576-mmi32.cn-hangzhou.nas.aliyuncs.com
            path: /
```

Dynamically provision a NAS volume

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  volumeClaimTemplates:
  - metadata:
      name: html
    spec:
      accessModes:
        - ReadWriteOnce
      storageClassName: alicloud-nas
      resources:
        requests:
          storage: 2Gi
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:alpine
        volumeMounts:
        - mountPath: "/usr/share/nginx/html/"
          name: html
```

6.4.4. Mount OSS volumes

You can mount Object Storage Service (OSS) buckets as volumes in Kubernetes clusters.

You can mount OSS buckets in the following ways:

- Mount an OSS bucket as a volume.
- Mount an OSS bucket by creating a PV and a PVC.

Prerequisites

To mount an OSS bucket as a statically provisioned volume, you must create an OSS bucket in the OSS console.

Background

- OSS buckets can be mounted only as statically provisioned volumes.
- An OSS bucket can be shared by multiple pods.
- bucket: Only buckets can be mounted to a Kubernetes cluster. The subdirectories or files in a bucket cannot be mounted to a Kubernetes cluster.
- url: specifies the endpoint of the OSS bucket. The endpoint is the domain name that is used to mount the OSS bucket to the Kubernetes cluster.
- akId: specifies your AccessKey ID.

- akSecret: specifies your AccessKey secret.
- otherOpts: the custom parameters that are used to mount the OSS bucket. The parameters must be in the following format: `-o *** -o ***`.

> **Note** To mount an OSS bucket as a volume, you must create a Secret with your AccessKey pair when you deploy the flexvolume Service.

Mount an OSS bucket as a statically provisioned volume

- **Mount an OSS bucket as a volume**

  Use the following *oss-deploy.yaml* file to create a 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"
```

- **Create a static PV and a PVC**

  i. **Create a PV**

  You can create a persistent volume (PV) in the Container Service console or by using a YAML file.

- **Create a PV by using a YAML file**

  Use the following *oss-pv.yaml* file to create a 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 a PV in the Container Service console**

  a. Log on to the Container Service console.

  b. In the left-side navigation pane, click **Clusters**.

  c. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.

  d. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.

  e. On the **Persistent Volumes** page, click **Create** in the upper-right corner.

f. In the Create PV dialog box, set the parameters.

Parameter	Description
PV Type	In this example, OSS is selected.
Name	Enter a name for the PV. The name must be unique in the cluster. In this example, pv-oss is used.
Volume Plug-in	By default, CSI is selected.
Capacity	Specify the capacity of the PV.
Access Mode	By default, ReadWriteMany is selected.
AccessKey ID and **AccessKey Secret**	The AccessKey pair that is required to access OSS buckets. To obtain your AccessKey pair, go to the Apsara Uni-manager Management Console, choose **Enterprise > Organizations**, click  on the right side of the organization. Then, click **AccessKey** and copy the AccessKey pair.
Optional Parameters	Enter custom parameters in the format of `-o *** -o ***`.
Bucket ID	Enter the name of the OSS bucket that you want to mount. Click **Select Bucket**. In the dialog box that appears, select the OSS bucket that you want to mount and click **Select**.
Endpoint	**Internal Endpoint** is recommended.
Label	Add labels to the PV.

g. Click **Create**.

ii. **Create a PVC**

Use the following *oss-pvc.yaml* file to create a persistent volume claim (PVC):

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

iii. **Create a pod**

Use the following *oss-pod.yaml* file to create a pod:

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

Can I mount an OSS bucket as a dynamically provisioned volume?

No. Dynamically provisioned OSS volumes are not supported.

6.4.5. Create a PVC

This topic describes how to create a persistent volume claim (PVC).

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A persistent volume (PV) is created. In this example, a PV created from a disk is used. For more information, see Use Apsara Stack disks.

  By default, PVCs are associated with PVs that have the alicloud-pvname label. This label is added to all PVs that are created in the Container Service console. If a PV does not have this label, manually add the label to the PV before you can associate the PV with a PVC.

Procedure

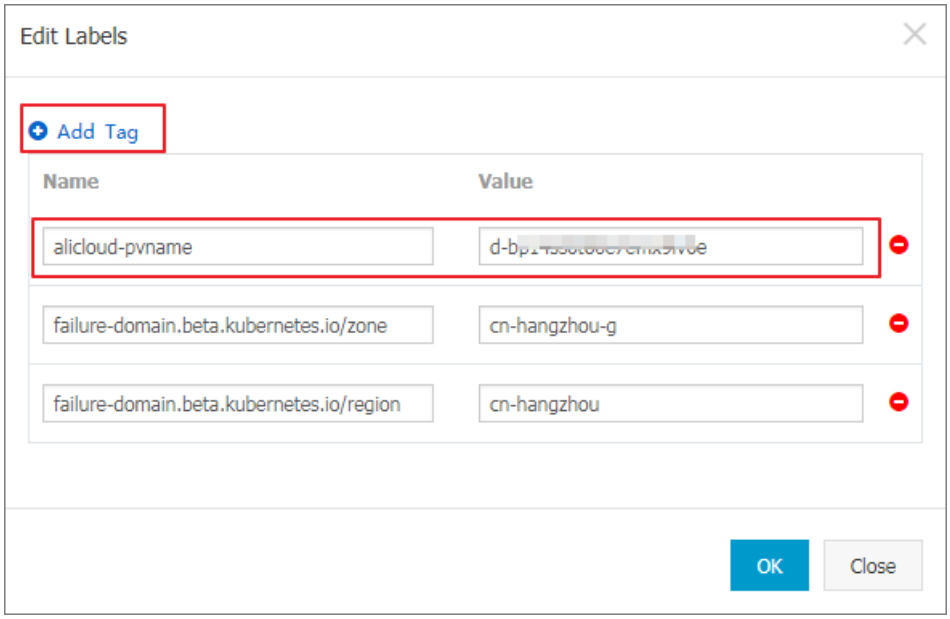
1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volume Claims**.
5. In the upper-right corner of the **Persistent Volume Claims** page, click **Create**.
6. In the **Create PVC** dialog box, set the parameters and click **Create**.
   - **PVC Type**: The PVC and PV must be of the same type. You can select Cloud Disk, NAS, and OSS.
   - **Name**: Enter the name of the PVC.
   - **Allocation Mode**: **Use StorageClass**, **Existing Volumes**, and **Create Volume** are supported.

In this example, **Use StorageClass** or **Existing Volumes** is selected.

- **Existing Storage Class**: Click **Select**. In the Select Storage Class dialog box, find the Storage Class that you want to use and click **Select** in the Actions column. This parameter is required only when you set Allocation Mode to **Use StorageClass**.
- **Existing Volumes**: Click **Select PV**. In the Select PV dialog box, find the PV that you want to use and click **Select** in the Actions column. This parameter is required only when you set Allocation Mode to **Existing Volumes**.
- **Capacity**: Specifies the capacity used by the PVC. The value cannot be larger than the total capacity of the associated PV.
- **Access Mode**: The default value is ReadWriteOnce. This parameter is required only when you set Allocation Mode to **Use StorageClass**.

> **Note** If your cluster has a PV that is not used, but you cannot find it in the **Select PV** dialog box, the reason may be that the PV does not have the alicloud-pvname label.

If you cannot find available PVs, click **Persistent Volumes** in the left-side navigation pane. On the Persistent Volumes page, find the PV that you want to use, click **Manage Labels** in the Actions column, and then add a label to the PV. On the Manage Labels dialog box, set the key of the label to alicloud-pvname and the value to the name of the PV. If the PV is created from a disk, the disk ID is used as the name of the PV.



7. Go to the Persistent Volume Claims page. You can find the newly created PVC in the list.

6.4.6. Use PVCs

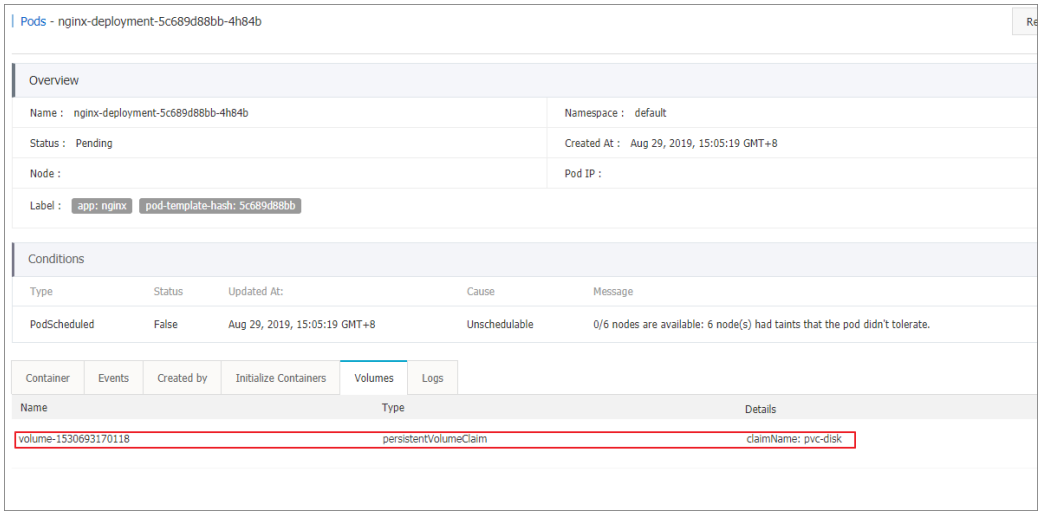
You can use persistent volume claims (PVCs) to mount volumes for applications.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A PVC is created. In this example, a PVC named pvc-disk is created to mount a disk volume. For more information, see Create PVCs.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click **Create from Image** in the upper-right corner.
6. On the **Basic Information** wizard page, specify the application name, number of replicas, type, and labels, select whether to synchronize the time zone from nodes to containers, and then click **Next**.
7. On the **Container** wizard page, select an image and configure volumes of the cloud storage type. You can select Cloud Disk, Apsara File Storage NAS (NAS), and Object Service Storage (OSS). In this example, select the PVC named pvc-disk and click **Next**. For more information, see Container configurations.
8. On the **Advanced** wizard page, create a Service for the test-nginx application and click **Create**.
9. On the **Complete** wizard page, click View Details to view detailed information about the application. You are redirected to the details page of the test-nginx application.
10. On the **Pods** tab, you can find the pods to which the application belongs. Select a pod and click **View Details** to view detailed information about the pod.
11. On the details page of the pod, click the **Volumes** tab. You can find that the pod is associated with the pvc-disk PVC.



6.5. Network management

6.5.1. Plan the network of a Container Service cluster

When you create a Container Service cluster, you must specify a virtual private cloud (VPC), vSwitches, the pod CIDR block, and the Service CIDR block. Therefore, we recommend that you plan the IP address of each Elastic Compute Service (ECS) instance, the pod CIDR block, and the Service CIDR block before you create the cluster. This topic describes how to plan CIDR blocks for a Container Service cluster that is deployed in a VPC and how each CIDR block is used.

VPC-related CIDR blocks and cluster-related CIDR blocks

Before you create a VPC, you must plan the CIDR block of the VPC and the CIDR blocks of vSwitches in the VPC. Before you create a Container Service cluster, you must plan the pod CIDR block and the Service CIDR block. Container Service supports the Terway and Flannel network plug-ins.

Use the Terway network plug-in

If your cluster uses the Terway network plug-in, you must configure the following parameters:

- VPC

  When you create a VPC, you must select a CIDR block for the VPC. The valid VPC CIDR blocks are 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.

- vSwitch

  The IP addresses of Elastic Compute Service (ECS) instances are assigned from vSwitches. This allows nodes to communicate with each other. The CIDR blocks of vSwitches in the VPC must be subsets of the VPC CIDR block. This means that the vSwitch CIDR blocks must be the same as or fall within the VPC CIDR block. Take note of the following items:

  - You must select vSwitches that belong to the VPC where the cluster resides.
  - The system allocates IP addresses from the CIDR block of a vSwitch to the ECS instances that are attached to the vSwitch.
  - You can create multiple vSwitches in a VPC. However, the CIDR blocks of these vSwitches cannot overlap with each other.
  - You must specify a pod vSwitch for each vSwitch in the same zone.

- Pod vSwitch

  The IP addresses of pods are assigned from the CIDR blocks of pod vSwitches. This allows pods to communicate with each other. A pod is a group of containers in a Kubernetes cluster. Each pod has an IP address. The CIDR blocks that you specify when you create pod vSwitches in the VPC must be subsets of the VPC CIDR block. Take note of the following items:

  - You must select vSwitches that belong to the VPC where the cluster resides.
  - In a Container Service cluster that has Terway installed, the IP addresses of pods are assigned by pod vSwitches.
  - The CIDR blocks of pod vSwitches cannot overlap with the CIDR blocks of vSwitches.
  - The CIDR blocks of pod vSwitches cannot overlap with the Service CIDR block.
  - You must specify a pod vSwitch for each vSwitch in the same zone.

- Service CIDR block

  The Service CIDR block provides IP addresses for ClusterIP type Services. Service is a Kubernetes concept. Each Service has an IP address. Take note of the following items:

  - The IP address of a Service is effective only within the cluster.
  - The Service CIDR block cannot overlap with the CIDR blocks of vSwitches.

- The Service CIDR block cannot overlap with the CIDR blocks of Pod vSwitches.

Use the Flannel network plug-in

If your cluster uses the Flannel network plug-in, you must configure the following parameters:

- VPC

  When you create a VPC, you must select a CIDR block for the VPC. The valid VPC CIDR blocks are 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.

- vSwitch

  The IP addresses of ECS instances are assigned from vSwitches. This allows nodes to communicate with each other. The CIDR blocks of vSwitches in the VPC must be subsets of the VPC CIDR block. This means that the vSwitch CIDR blocks must be the same as or fall within the VPC CIDR block. Take note of the following items:

  - You must select vSwitches that belong to the VPC where the cluster resides.
  - The system allocates IP addresses from the CIDR block of a vSwitch to the ECS instances that are attached to the vSwitch.
  - You can create multiple vSwitches in a VPC. However, the CIDR blocks of these vSwitches cannot overlap with each other.

- Pod CIDR block

  The IP addresses of pods are allocated from the pod CIDR block. This allows pods to communicate with each other. A pod is a group of containers in a Kubernetes cluster. Each pod has an IP address. Take note of the following items:

  -
  - The pod CIDR block cannot overlap with the CIDR blocks of vSwitches.
  - The pod CIDR block cannot overlap with the Service CIDR block.

  For example, if the VPC CIDR block is 172.16.0.0/12, the pod CIDR block cannot be 172.16.0.0/16 or 172.17.0.0/16, because these CIDR blocks are subsets of 172.16.0.0/12.

- Service CIDR block

  The Service CIDR block provides IP addresses for ClusterIP type Services. Service is a Kubernetes concept. Each Service has an IP address. Take note of the following items:

  - The IP address of a Service is effective only within the cluster.
  - The Service CIDR block cannot overlap with the CIDR blocks of vSwitches.
  - The Service CIDR block cannot overlap with the CIDR blocks of Pod vSwitches.

Plan the cluster network

To use Container Service clusters that are deployed on Alibaba Cloud, you must first set up networks for the clusters based on the cluster sizes and business scenarios. You can refer to the following tables to set up networks for Kubernetes clusters. Change specifications as needed in unspecified scenarios.

- Plane the VPC

Number of nodes	Requirement	Number of VPCs	Number of zones
< 100	Regular business	A single VPC	1

Number of nodes	Requirement	Number of VPCs	Number of zones
Arbitrary	Cross-zone deployment	A single VPC	≥ 2
Arbitrary	High reliability and cross-region deployment	Multiple VPCs	≥ 2

- Plan CIDR blocks for clusters

  The following tables describe how to plan CIDR blocks for clusters that use Flannel or Terway.

  - **Clusters that use Flannel**

VPC CIDR block	vSwitch CIDR block	Pod CIDR block	Service CIDR block	Maximum number of pod IP addresses
192.168.0.0/16	192.168.0.0/24	172.20.0.0/16	172.21.0.0/20	65536

  - **Clusters that use Terway**

    - **Terway mode**

VPC CIDR block	vSwitch CIDR block	Pod vSwitch CIDR block	Service CIDR block	Maximum number of pod IP addresses
192.168.0.0/16	192.168.0.0/19	192.168.32.0/19	172.21.0.0/20	8192

    - **Multi-zone deployment**

VPC CIDR block	vSwitch CIDR block	Pod vSwitch CIDR block	Service CIDR block	Maximum number of pod IP addresses
192.168.0.0/16	Zone I 192.168.0.0/19	192.168.32.0/19	172.21.0.0/20	8192
	Zone J 192.168.64.0/19	192.168.96.0/19		8192

How to plan CIDR blocks

- **Scenario 1: One VPC and one Kubernetes cluster**

  This is the simplest scenario. The CIDR block of a VPC is specified when you create the VPC. When you create a cluster in the VPC, make sure that the pod CIDR block and the Service CIDR block do not overlap with the VPC CIDR block.

- **Scenario 2: One VPC and multiple Kubernetes clusters**

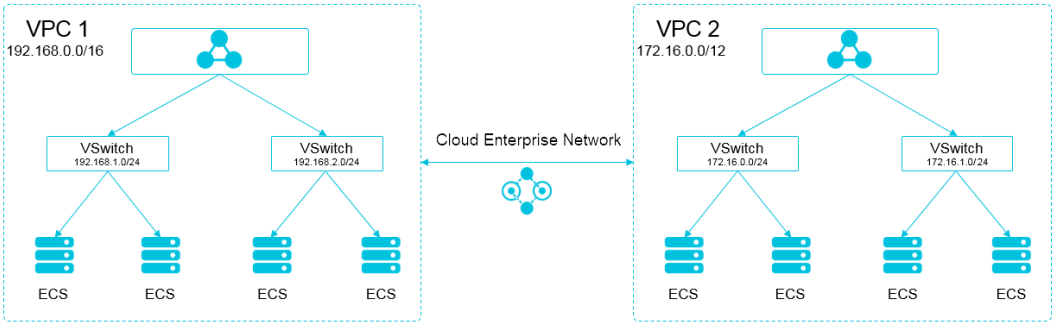
  You want to create more than one cluster in a VPC.

- The CIDR block of the VPC is specified when you create the VPC. When you create clusters in the VPC, make sure that the VPC CIDR block, Service CIDR block, and pod CIDR block of each cluster do not overlap with one another.
- The Service CIDR blocks of the clusters can overlap with each other. However, the pod CIDR blocks cannot overlap with each other.
- In the default network mode (Flannel), the packets of pods must be forwarded by the VPC router. Container Service automatically generates a route table for each destination pod CIDR block on the VPC router.

> **Note** In this case, a pod in one cluster can communicate with the pods and ECS instances in another cluster. However, the pod cannot communicate with the Services in another cluster.

- **Scenario 3: Two connected VPCs**

  If two VPCs are connected, you can use the route table of one VPC to specify the packets that you want to send to the other VPC. The CIDR block of VPC 1 is 192.168.0.0/16 and the CIDR block of VPC 2 is 172.16.0.0/12, as shown in the following figure. You can use the route table of VPC 1 to forward all packets that are destined for 172.16.0.0/12 to VPC 2.



Connected VPCs

Item	IP address/CIDR block	Destination CIDR block	Destination VPC
VPC 1	192.168.0.0/16	172.16.0.0/12	VPC 2
VPC 2	172.16.0.0/12	192.168.0.0/16	VPC 1

In this scenario, make sure that the following conditions are met when you create a cluster in VPC 1 or VPC 2:

- The CIDR blocks of the cluster do not overlap with the CIDR block of VPC 1.
- The CIDR blocks of the cluster do not overlap with the CIDR block of VPC 2.
- The CIDR blocks of the cluster do not overlap with those of other clusters in VPC 1 and VPC 2.
- The CIDR blocks of the cluster do not overlap with those of pods in VPC 1 and VPC 2.
- The CIDR blocks of the cluster do not overlap with those of Services in VPC 1 and VPC 2.

In this example, you can set the pod CIDR block of the cluster to a subset of 10.0.0.0/8.

> **Note** All IP addresses in the destination CIDR block of VPC 2 can be considered in use. Therefore, the CIDR blocks of the cluster cannot overlap with the destination CIDR block.

To access pods in VPC 1 from VPC 2, you must configure a route in VPC 2. The route must point to the pod CIDR block of a cluster in VPC 1.

- **Scenario 4: One VPC and one data center**

  If a VPC is connected to a data center, packets of specific CIDR blocks are routed to the data center. In this case, the pod CIDR block of a cluster in the VPC cannot overlap with these CIDR blocks. To access pods in the VPC from the data center, you must configure a route in the data center to enable VBR-to-VPC peering connection.

6.5.2. Set access control for pods

This topic describes how to use network policies to control access between pods.

Prerequisites

You have created a Kubernetes cluster and selected the **Terwaynetwork plug-in**. For more information, see Create a Kubernetes cluster.

Context

You can declare network policies to control access between pods and thus prevent applications from interfering each other.

Procedure

For more information about standard Kubernetes network policies, see Network policies.

1. Create a pod that runs as a server and attach `label run=nginx` to the pod. For more information, see Create an application from an orchestration template.

   The sample YAML file is as follows:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
     name: server
     labels:
       run: nginx
   spec:
     containers:
       - name: nginx
         image: registry.acs.intranet.env22.com/nginx:1.8
   ```

2. Create a network policy. For more information, see Create an application from an orchestration template.

   The sample YAML file is as follows:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: access-nginx
spec:
  podSelector:
    matchLabels:
      run: nginx # Apply the network policy to pods with the run=nginx label
  ingress:
  - from:
    - podSelector:
        matchLabels:
          access: "true" # Only pods with the access=true label are accessible
```

3. Use the *client.yaml* and *client-label* files to create two pods that run as clients.

   One pod has the required label and the other does not.

   i. Create the *client.yaml* and *client-label* files with the following contents respectively.

   ```
   # This pod has no label
   apiVersion: v1
   kind: Pod
   metadata:
     name: client
   spec:
     containers:
       - name: busybox
         image: registry.acs.intranet.env22.com/acs/busybox
         command: ["sh", "-c", "sleep 200000"]
   ```

   ```
   # This pod has the label
   apiVersion: v1
   kind: Pod
   metadata:
     name: client-label
     labels:
       access: "true"
   spec:
     containers:
       - name: busybox
         image: registry.acs.intranet.env22.com/acs/busybox
         command: ["sh", "-c", "sleep 200000"]
   ```

   ii. Run the following commands to create these pods:

   ```
   kubectl apply -f client.yaml
   kubectl apply -f client-label.yaml
   ```

   You can see that only the pod with the required label can access the server.

6.5.3. Set bandwidth limits for pods

This topic describes how to limit the bandwidth of inbound and outbound traffic that flows through a pod.

Prerequisites

You have created a Kubernetes cluster and selected the **Terwaynetwork plug-in**. For more information, see Create a Kubernetes cluster.

Context

Throttling pods helps prevent performance degradation of the host or other workloads when certain pods occupy excessive resources.

Method

You can use the `k8s.aliyun.com/ingress-bandwidth` and `k8s.aliyun.com/egress-bandwidth` annotations for pod throttling.

- `k8s.aliyun.com/ingress-bandwidth`: limits the pod inbound bandwidth.
- `k8s.aliyun.com/egress-bandwidth`: limits the pod outbound bandwidth.
- The bandwidth limit is measured in m and k, which represent Mbit/s and Kbit/s respectively.

Procedure

1. Create a pod that runs as a server in the console. For more information, see Create an application from an orchestration template.

   The sample YAML file is as follows:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
     name: server
     annotations:
       k8s.aliyun.com/ingress-bandwidth: 10m # Set the inbound bandwidth limit to 10 Mbit/
   s
       k8s.aliyun.com/egress-bandwidth: 10m
   spec:
     containers:
       - name: nginx
         image: registry.acs.intranet.env22.com/nginx:1.8
   ```

2. Run the kubectl exec command to connect to the pod. To verify that pod throttling is effective, run the following commands to create a file on the pod. Assume that the IP address of the pod created in step 1 is 172.16.XX.XX.

   ```
   cd /usr/share/nginx/html
   dd if=/dev/zero of=bigfile bs=1M count=1000
   ```

3. Use the *client-deploy.yaml* file to create a pod that runs as a client.

i. Create the *client-deploy.yaml* file with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: client
  annotations:
     k8s.aliyun.com/ingress-bandwidth: 10m # Set the inbound bandwidth limit to 10
Mbit/s
     k8s.aliyun.com/egress-bandwidth: 10m
spec:
  containers:
    - name: busybox
      image: registry.acs.intranet.env22.com/acs/netdia
      command: ["sh", "-c", "sleep 200000"]
```

ii. Run the following command to create the pod:

```
kubectl apply -f client-deploy.yaml
```

4. Run the following command to check whether bandwidth is limited:

```
kubectl exec -it client sh
```

6.5.4. Work with Terway

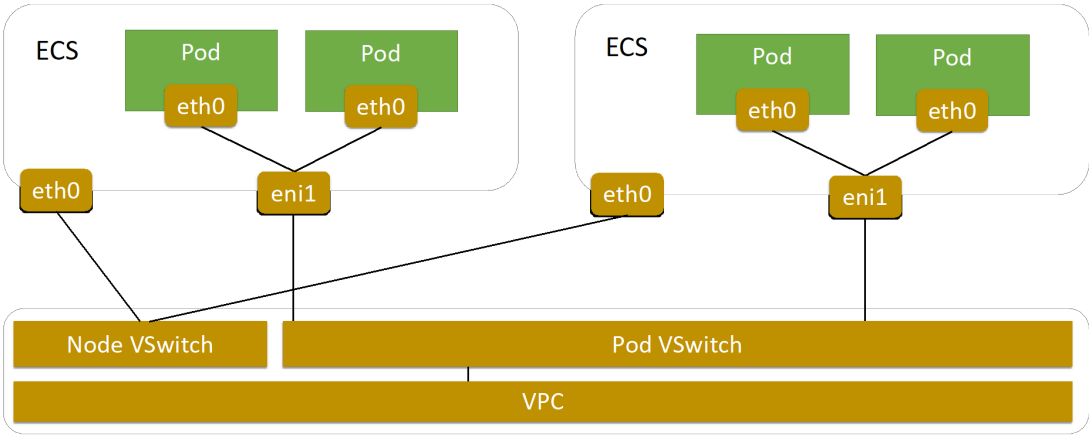
Terway is an open source Container Network Interface (CNI) plug-in developed by Alibaba Cloud. Terway works with Virtual Private Cloud (VPC) and allows you to use standard Kubernetes network policies to regulate how containers communicate with each other. You can use Terway to enable internal communication within a Kubernetes cluster. This topic describes how to use Terway.

Context

Terway is a network plug-in developed by Alibaba Cloud for Container Service. Terway allows you to set up networks for pods by associating Alibaba Cloud elastic network interfaces (ENIs) with the pods. Terway also allows you to use standard Kubernetes network policies to regulate how containers communicate with each other. In addition, Terway is compatible with Calico network polices.

In a cluster that has Terway installed, each pod has a separate network stack and is assigned a separate IP address. Pods on the same Elastic Compute Service (ECS) instance communicate with each other by forwarding packets inside the ECS instance. Pods on different ECS instances communicate with each other through ENIs in the VPC where the ECS instances are deployed. This improves communication efficiency because no tunneling technologies such as Virtual Extensible Local Area Network (VXLAN) are required to encapsulate packets.

How Terway works



Comparison between Flannel and Terway

When you create a Kubernetes cluster, you can choose one of the following network plug-ins:

- Terway: a network plug-in developed by Alibaba Cloud for Container Service. Terway allows you to assign ENIs to containers and use standard Kubernetes network policies to regulate how containers communicate with each other. Terway also supports bandwidth throttling on individual containers. Terway uses flexible IP Address Management (IPAM) policies to allocate IP addresses to containers. This avoids IP address waste. If you do not want to use network policies, you can select Flannel as the network plug-in. Otherwise, we recommend that you select Terway.
- Flannel: an open source CNI plug-in, which is simple and stable. You can use Flannel with VPC of Alibaba Cloud. This allows your clusters and containers to run in high-performance and stable networks. However, Flannel provides only basic features. It does not support standard Kubernetes network policies. For more information, see Flannel.

Item	Terway	Flannel
Performance	The IP address of each pod in a Kubernetes cluster is assigned from the CIDR block of the VPC where the cluster is deployed. Therefore, you do not need to use the Network Address Translation (NAT) service to translate IP addresses. This avoids IP address waste. In addition, each pod in the cluster can use an exclusive ENI.	Flannel works with VPC of Alibaba Cloud. The CIDR block of pods that you specify must be different from that of the VPC where the cluster is deployed. Therefore, the NAT service is required and some IP addresses may be wasted.
Security	Terway supports network policies.	Flannel does not support network policies.
IP address management	Terway allows you to assign IP addresses on demand. You do not have to assign CIDR blocks by node. This avoids IP address waste.	You can only assign CIDR blocks by node. In large-scale clusters, a great number of IP addresses may be wasted.

Item	Terway	Flannel
SLB	Server Load Balancer (SLB) directly forwards network traffic to pods. You can upgrade the pods without service interruption.	SLB forwards network traffic to the NodePort Service. Then, the NodePort Service routes the network traffic to pods.

Considerations

- To use the Terway plug-in, we recommend that you use ECS instances of higher specifications and newer types, such as ECS instances that belong to the g5 or g6 instance family with at least eight CPU cores. For more information, see the *Instance families* chapter of *ECS User Guide*.
- The maximum number of pods that each node supports is based on the number of ENIs assigned to the node.
  - Maximum number of pods supported by each shared ENI = (Number of ENIs supported by each ECS instance - 1) × Number of private IP addresses supported by each ENI
  - Maximum number of pods supported by each exclusive ENI = Number of ENIs supported by each ECS instance -1

Step 1: Plan CIDR blocks

When you create a Kubernetes cluster, you must specify a VPC, vSwitches, the CIDR block of pods, and the CIDR block of Services. If you want to install the Terway plug-in, you must first create a VPC and two vSwitches in the VPC. The two vSwitches must be created in the same zone.

You can assign the following CIDR blocks for a cluster that uses Terway.

- VPC CIDR block: 192.168.0.0/16
- vSwitch CIDR block: 192.168.0.0/19
- CIDR block of pod vSwitch: 192.168.32.0/19
- Service CIDR block:172.21.0.0/20

> **Note**
> - IP addresses within the CIDR block of the vSwitch are assigned to nodes.
> - IP addresses within the CIDR block of the pod vSwitch are assigned to pods.

The following example describes how to create a VPC and two vSwitches. The CIDR blocks in the preceding section are assigned in this example.

1. Log on to the VPC console.

   > **Note** For more information, see the *Log on to the VPC console* chapter of *VPC*.

2. Create a VPC.
   i. In the top navigation bar, select the region where you want to create the VPC.
   ii. On the **VPCs** page, click **Create VPC**.
   iii. On the **Create VPC** page, set the following parameters and click **Submit**.

Parameter	Description
Organization	Select the organization to which the VPC belongs.
Resource Set	Select the resource set to which the VPC belongs.
Region	Select the region where you want to deploy the VPC.
Sharing Scope	Select the sharing scope of the VPC. ■ **Current Resource Set**: Only the administrator of the current resource set can use the VPC to create resources. ■ **Current Organization and Subordinate Organization**: Only the administrators of the current organization and its subordinate organization can create resources for the shared VPC. ■ **Current Organization**: Only the administrator of the current organization can use create resources for the shared VPC.
VPC Name	Enter a name for the VPC. In this example, vpc_192_168_0_0_16 is used. The name must be 2 to 128 characters in length, and can contain letters, digits, underscores (_), and hyphens (-). The name must start with a letter and cannot start with `http://` or `https://`.
IPv4 CIDR Block	Select an IPv4 CIDR block for the VPC. In this example, 192.168.0.0/16 is specified. **Note** After a VPC is created, the IPv4 CIDR block of the VPC cannot be modified.
IPv6 CIDR Block	Specify whether to assign an IPv6 CIDR block. ■ **Do Not Assign**: The system does not assign an IPv6 CIDR block to the VPC. ■ **Assign**: An IPv6 CIDR block is automatically assigned to the VPC. If you set this parameter to Assign, the system automatically creates a free IPv6 gateway for this VPC, and assigns an IPv6 CIDR block with the subnet mask /56, such as 2xx1: db8::/56. By default, IPv6 addresses can be used to communicate within only private networks. If you want to allow an instance assigned with an IPv6 address to access the Internet or be accessed by IPv6 clients over the Internet, you must purchase an Internet bandwidth plan for the IPv6 address. For more information, see the **Activate IPv6 Internet bandwidth** section of the **Manage IPv6 Internet bandwidth** topic of the *IPv6 gateway user guide*.
Description	Enter a description for the VPC. The description must be 2 to 256 characters in length, and can contain digits, underscores (_), hyphens (-), periods (.), colons (:), and commas (,). It must start with a letter and cannot start with `http://` or `https://`.

3. Create vSwitches.

i. In the left-side navigation pane, click **VSwitches**.
ii. Select the region of the VPC in which you want to create a vSwitch.
iii. On the **VSwitches** page, click **Create VSwitch**.
iv. On the **Create vSwitch** page, set the following parameters and click **Submit**.

> **Note** Make sure that the two vSwitches are in the same zone.

Parameter	Description
Organization	Select the organization to which the vSwitch belongs.
Resource Set	Select the resource set to which the vSwitch belongs.
Region	Select the region where you want to deploy the vSwitch.
Zone	Select the zone to which the vSwitch belongs. In a VPC, each vSwitch can be deployed in only one zone. You cannot deploy a vSwitch across zones. However, you can deploy cloud resources in vSwitches that belong to different zones to achieve cross-zone disaster recovery. **Note** A cloud instance can be deployed in only one vSwitch.
Sharing Scope	Select the sharing scope of the vSwitch. ■ **Current Resource Set**: Only the administrator of the current resource set can create resources in the shared vSwitch. ■ **Current Organization and Subordinate Organization**: Only the administrators of the current organization and its subordinate organizations can create resources in the shared vSwitch. ■ **Current Organization**: Only the administrator of the current organization can create resources in the shared vSwitch.
VPC	Select the VPC for which you want to create the vSwitch.
Dedicated for Out-of-cloud Physical Machines	Specify whether the vSwitch to be created is dedicated for bare-metal instances. For more information about bare-metal instances, see the **VPC bare-metal instance features** topic in *BMS User Guide*.
vSwitch Name	Enter a name for the vSwitch. In this example, node_switch_192_168_0_0_19 is used. The name must be 2 to 128 characters in length and can contain digits, underscores (_), hyphens (-), periods (.), colons (:), and commas (,). It must start with a letter and cannot start with `http://` or `https://`.
IPv4 CIDR Block	Specify an IPv4 CIDR block for the vSwitch. In this example, 192.168.0.0/19 is used.

Parameter	Description
IPv6 CIDR Block	Specify an IPv6 CIDR block for the vSwitch. ■ You must check whether IPv6 is enabled for the specified VPC. If IPv6 is disabled, you cannot assign an IPv6 CIDR block to the vSwitch. ■ If IPv6 is enabled, you can enter a decimal number ranging from 0 to 255 to define the last 8 bits of the IPv6 CIDR block of the vSwitch. For example, if the IPv6 CIDR block of the VPC is 2xx1:db8::/64, specify 255 to define the last 8 bits of the IPv6 CIDR block. In this case, the IPv6 CIDR block of the vSwitch is 2xx1:db8:ff::/64. ff is the hexadecimal value of 255.
Description	Enter a description for the vSwitch. The description must be 2 to 256 characters in length and can contain letters, digits, underscores (_), hyphens (-), periods (.), colons (:), and commas (,). The name must start with a letter and cannot start with `http://` or `https://`.

4. Repeat Step to create a pod vSwitch. Set the name of the pod vSwitch to **pod_switch_192_168_32_0_19** and IPv4 CIDR Block to **192.168.32.0/19**.

Step 2: Set up networks for a cluster that uses Terway

To install Terway in a cluster and set up networks for the cluster, set the following parameters.

> **Note** A Kubernetes cluster is used as an example to show how to set up networks for a cluster that uses Terway as the network plug-in. For more information about how to create a Kubernetes cluster, see Create a Kubernetes cluster.

- **VPC**: Select the VPC created in Step 1: Plan CIDR blocks.
- **VSwitch**: Select the vSwitch created in Step 1: Plan CIDR blocks.
- **Network Plug-in**: Select **Terway**.
- **Pod VSwitch**: Select the pod vSwitch created in Step 1: Plan CIDR blocks.
- **Service CIDR**: Use the default settings.

6.6. Namespaces

6.6.1. Create a namespace

You can create a namespace in the Container Service console.

Prerequisites

A Kubernetes cluster is created.

Context

In a Kubernetes cluster, you can use namespaces to create multiple virtual spaces. When a large number of users share a cluster, you can use namespaces to divide different workspaces and allocate cluster resources to different tasks. You can also use resource quotas to allocate resources to each namespace.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, click **Namespaces and Quotas**.
5. On the **Namespace** page, click **Create** in the upper-right corner.
6. In the **Create Namespace** dialog box, configure the namespace.

Create a namespace

Parameter	Description
Name	Enter a name for the namespace. In this example, test is entered. The name must be 1 to 63 characters in length and can contain digits, letters, and hyphens (-). It must start and end with a letter or digit.
Label	**Label**: Add one or more labels to the namespace. Labels are used to identify namespaces. For example, you can label a namespace as one used in the test environment. To add a label, enter a key and a value and click **Add** in the Actions column.

7. Click **OK**.

   You can find the namespace that you created on the Namespace page.

6.6.2. Set resource quotas and limits

You can set resource quotas and limits for a namespace in the Container Service console.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A sample namespace named **test** is created. For more information, see Create a namespace.
- You are connected to a master node of the cluster. For more information, see Connect to a Kubernetes cluster through kubectl.

Context

By default, a running pod uses the CPU and memory resources of a node without limit. This means that pods can compete for computing resources of a cluster. As a result, the pods in a namespace may exhaust all of the computing resources.

Namespaces can be used as virtual clusters to serve multiple purposes. We recommend that you set resource quotas for namespaces.

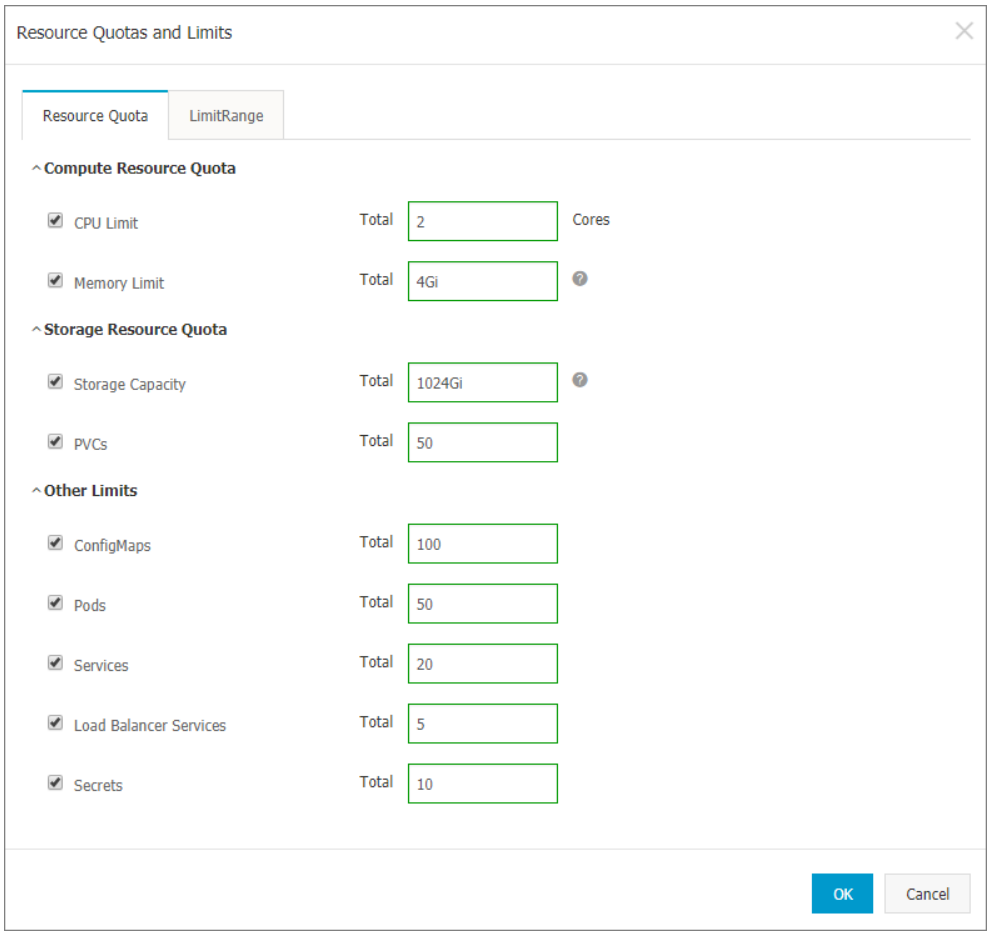
For a namespace, you can set quotas on resources such as CPU, memory, and pod quantity. For more information, see Resource Quotas.

Procedure

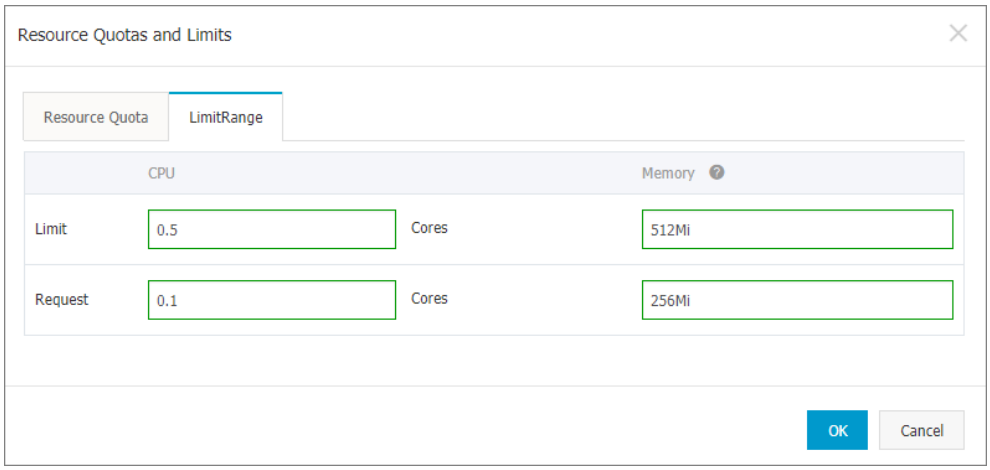
1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, click **Namespaces and Quotas**.
5. Find the namespace that you want to manage and click **Resource Quotas and Limits** in the **Actions** column.
6. In the dialog box that appears, set resource quotas and default resource limits.

> **Note** After you set CPU and memory quotas for a namespace, you must specify CPU and memory limits when you create a pod. You can also set the default resource limits for the namespace. For more information, see Resource Quotas.

i. Set resource quotas for the namespace.



ii. You can set resource limits and resource requests for containers in the namespace. This enables you to control the amount of resources consumed by the containers. For more information, see https://kubernetes.io//memory-default-namespace/.



7. After you set resource quotas and limits, connect to a master node of the cluster and run the following commands to query the resource configurations of the namespace.

```
# kubectl get limitrange,ResourceQuota -n test
NAME AGE
limitrange/limits 8m
NAME AGE
resourcequota/quota 8m
# kubectl describe limitrange/limits resourcequota/quota -n test
Name: limits
Namespace: test
Type Resource Min Max Default Request Default Limit Max Limit/Request Ratio
---- -------- --- --- --------------- ------------- -----------------------
Container cpu - - 100m 500m -
Container memory - - 256Mi 512Mi -
Name: quota
Namespace: test
Resource Used Hard
-------- ---- ----
configmaps 0 100
limits.cpu 0 2
limits.memory 0 4Gi
persistentvolumeclaims 0 50
pods 0 50
requests.storage 0 1Ti
secrets 1 10
services 0 20
services.loadbalancers 0 5
```

6.6.3. Modify a namespace

You can modify an existing namespace.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A sample namespace named **test** is created. For more information, see Create a namespace.

Context

When you modify a namespace, you can add, delete, or modify the labels that are added to the namespace based on your requirements.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, click **Namespaces and Quotas**.
5. Find the namespace that you want to modify and click **Edit** in the **Actions** column.
6. In the **Edit Namespace** dialog box, find the label that you want to modify and click **Edit** to modify the label. For example, you can change the key-value pair of the label to `env:test-V2`. Then, click **Save**.

7. Click **OK**.

   You can find that the labels of the namespace on the Namespace page are updated.

6.6.4. Delete a namespace

You can delete namespaces that are no longer in use.

Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- A sample namespace named **test** is created. For more information, see Create a namespace.

Context

> **Note** When you delete a namespace, all resource objects under the namespace are deleted. Proceed with caution.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, click **Namespaces and Quotas**.
5. Find the namespace that you want to delete and click **Delete** in the **Actions** column.
6. In the message that appears, click **Confirm**.

   Return to the Namespace page. You can find that the namespace is deleted. Resource objects in the namespace are also deleted.

6.7. Applications

6.7.1. Create a stateless application by using a Deployment

You can deploy a stateless application by using a Deployment. A Deployment can be created by using an image, an orchestration template, or kubectl commands. This topic describes how to create a stateless NGINX application by using an image, an orchestration template, and kubectl commands.

Prerequisites

- A Container Service cluster is created. For more information, see Create a Kubernetes cluster.
- A kubectl client is connected to the cluster. For more information, see Connect to a cluster through kubectl.
- A private image repository is created and your image is uploaded to the repository.

Create a Deployment from an image

Step 1: Configure basic settings

1. Log on to the Container Service console
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**.
5. In the upper part of the **Deployments** page, select the namespace and click **Create from Image**.
6. On the **Basic Information** wizard page, configure the basic settings of the application.

   Before you configure the Deployment, select a namespace in the upper part of the page. In this example, the **default** namespace is selected.

Parameter	Description
Name	Enter a name for the application.
Replicas	The number of pods that you want to provision for the application. Default value: 2.
Type	The type of the application. You can select **Deployment**, **StatefulSet**, **Job**, **CronJob**, or **DaemonSet**.
Label	Add labels to the application. The labels are used to identify the application.
Annotations	Add annotations to the application.
Synchronize Timezone	Specify whether to synchronize the time zone between nodes and containers.

   > **Note** In this example, the **Deployment** type is selected.

7. Click **Next**.

   Proceed to the **Container** wizard page.

Step 2: Configure containers

On the **Container** wizard page, configure the following container settings: the container image, resource request and limit, container ports, environment variables, health check settings, lifecycle configurations, volumes, and logging configurations.

> **Note** On the **Container** wizard page, Click **Add Container** to add more containers to the pod.

1. In the **General** section, configure the basic settings of the container.

Parameter	Description
Image	To select an image, click **Select Image**. In the upper right corner of the Select Image dialog box, select **Default** or **enterprise-edition** from the **Container Registry Instance** drop-down list. Then, select the image that you want to use and click **OK**. In this example, the nginx image is selected. **Note** ◦ Default: shows images that are stored on the Container Registry Standard Edition instance. ◦ enterprise-edition: shows images that are stored on the Container Registry Advanced Edition instance. You can also enter the address of a private image registry. Specify a private registry in the following format: `domainname/namespace/imagename`.
Image Version	◦ Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used. ◦ You can select the following image pulling policies: ■ **ifNotPresent**: If the image that you want to pull is found on your on-premises machine, the image on your on-premises machine is used. Otherwise, Container Service pulls the image from the image registry. ■ **Always**: Container Service pulls the image from the registry each time the application is deployed or expanded. ■ **Never**: Container Service uses only images on your on-premise machine. **Note** If you select **Image Pull Policy**, no image pulling policy is applied. ◦ To pull the image without a password, click **Set Image Pull Secret** to configure a Secret for pulling images.
Resource Limit	You can specify an upper limit for the CPU, memory, and ephemeral storage space that the container can consume. This prevents the container from occupying an excessive amount of resources.
Required Resources	The amount of the CPU resources, memory resources, and ephemeral storage space that are reserved for this application. These resources are exclusive to the container. This prevents the application from becoming unavailable when other services or processes compete for computing resources.

Parameter	Description
Container Start Parameter	◦ stdin: specifies that start parameters are sent to the container as standard input (stdin). ◦ tty: specifies that start parameters defined in a virtual terminal are sent to the container. The two options are usually used together. In this case, the virtual terminal (tty) is associated to the stdin of the container. For example, an interactive program receives the stdin from the user and displays the content in the terminal.
Privileged Container	◦ If you select Privileged Container, privileged=true is specified for the container and the privilege mode is enabled. ◦ If you do not select Privileged Container, privileged=false is specified for the container and the privilege mode is disabled.
Init Container	If you select Init Container, an init container is created. An init container provides tools for pod management. For more information, see init containers.

2. (Optional)In the **Ports** section, click **Add** to configure one or more container ports.

Parameter	Description
Name	Enter a name for the port.
Container Port	The container port that you want to open. Valid values: 1 to 65535.
Protocol	Valid values: TCP and UDP.

3. (Optional)In the **Environments** section, click **Add** to set environment variables.

   You can configure environment variables in key-value pairs for pods. Environment variables are used to apply pod configurations to containers. For more information, see Pod variables.

Parameter	Description
Type	Select the type of environment variable. Valid values: ◦ **Custom** ◦ **ConfigMaps** ◦ **Secret** ◦ **Value/ValueFrom** ◦ **ResourceFieldRef**

Parameter	Description
Variable	Specify the name of the environment variable.
Value/ValueFrom	Specify a reference that is used to define the environment variable.

4. (Optional)In the **Health Check** section, you can enable **liveness** and **readiness** probes as needed.

   For more information, see Configure Liveness, Readiness and Startup Probes.

Parameter	Request type	Configuration description
	HTTP	Sends an HTTP GET request to the container. You can set the following parameters: ○ Protocol: HTTP or HTTPS. ○ Path: the requested path on the server. ○ Port: Enter the container port that you want to open. Valid values: 1 to 65535. ○ HTTP Header: the custom headers in the HTTP request. Duplicate headers are allowed. You can set HTTP headers in key-value pairs. ○ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the wait time (in seconds) before the first probe is performed after the container is started. Default value: 3. ○ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ○ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ○ Healthy Threshold: the minimum number of consecutive successes that must occur before a container is considered healthy after a failed probe. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ○ Unhealthy Threshold: the minimum number of consecutive failures that must occur before a container is considered unhealthy after a success. Default value: 3. Minimum value: 1.

Parameter	Request type	Configuration description
○ Liveness: Liveness probes are used to determine when to restart the container. ○ Readiness: Readiness probes are used to determine whether the container is ready to accept traffic.	TCP	Sends a TCP socket to the container. kubelet attempts to open the socket on the specified port. If the connection can be established, the container is considered healthy. Otherwise, the container is considered unhealthy. You can configure the following parameters: ○ Port: Enter the container port that you want to open. Valid values: 1 to 65535. ○ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the wait time (in seconds) before the first probe is performed after the container is started. Default value: 15. ○ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ○ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ○ Healthy Threshold: the minimum number of consecutive successes that must occur before a container is considered healthy after a failed probe. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ○ Unhealthy Threshold: the minimum number of consecutive failures that must occur before a container is considered unhealthy after a success. Default value: 3. Minimum value: 1.

Parameter	Request type	Configuration description
	Command	Runs a probe command in the container to check the health status of the container. You can configure the following parameters: ◦ Command: the probe command that is run to check the health status of the container. ◦ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the wait time (in seconds) before the first probe is performed after the container is started. Default value: 5. ◦ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ◦ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ◦ Healthy Threshold: the minimum number of consecutive successes that must occur before a container is considered healthy after a failed probe. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ◦ Unhealthy Threshold: the minimum number of consecutive failures that must occur before a container is considered unhealthy after a success. Default value: 3. Minimum value: 1.

5. (Optional)In the **Lifecycle** section, configure the lifecycle of the container.

   You can set the following parameters to configure the lifecycle of the container: Start, Post Start, and Pre Stop. For more information, see Configure the lifecycle of a container.

Parameter	Description
Start	Specify a command and parameter that take effect before the container starts.
Post Start	Specify a command that takes effect after the container starts.
Pre Stop	Specify a command that takes effect before the container stops.

6. (Optional)In the **Volume** section, you can mount local volumes and persistent volume claims (PVCs) to the container.

Parameter	Description
Add Local Storage	You can select HostPath, ConfigMap, Secret, or EmptyDir from the PV Type drop-down list. Then, set Mount Source and Container Path to mount the volume to the container. For more information, see Volumes.
Add PVC	You can mount persistent volumes (PVs) by using PVCs. You must create a PVC before you can select the PVC to mount a PV. For more information, see Create a PVC.

7. (Optional)In the **Log** section, you can specify logging configurations and add custom tags to the collected log.

   Notice Make sure that the Log Service agent is installed in the cluster.

Parameter	Description
Collection Configuration	Logstore: creates a Logstore in Log Service to store the collected log.
	Log Path in Container: Specify stdout or a container path to collect log data. ○ Collect stdout files: If you specify stdout, the stdout files are collected. ○ Text Logs: specifies that the log in the specified path of the container is collected. In this example, */var/log/nginx* is specified as the path. Wildcard characters can be used in the path.
Custom Tag	You can also add custom tags. The tags are added to the log of the container when the log is collected. Custom tags provide an easy method to filter collected the log and perform statistical analysis.

8. Click **Next**.

   Proceed to the **Advanced** wizard page.

Step 3: Configure advanced settings

On the **Advanced** wizard page, configure the following settings: access control, scaling, scheduling, annotations, and labels.

1. In the **Access Control** section, you can configure access control settings for exposing backend pods.

> **Note**
>
> You can configure the following access control settings based on your business requirements:
>
> - Internal applications: For applications that provide services within the cluster, you can create a **ClusterIP** or **NodePort** Service to enable internal communication.
> - External applications: For applications that are exposed to the Internet, you can configure access control by using one of the following methods:
>   - Create a LoadBalancer Service. When you create a Service, set Type to **Server Load Balancer**. You can select or create a Server Load Balancer (SLB) instance for the Service and use the Service to expose your application to the Internet.
>   - Create an Ingress and use it to expose your application to the Internet. For more information, see Ingress.

You can also specify how the backend pods are exposed to the Internet. In this example, a ClusterIP Service and an Ingress are created to expose the NGINX application to the Internet.

- Click **Create** to the right side of **Services**. In the Create Service dialog box, configure the following parameters.

Parameter	Description
Name	Enter a name for the Service.

Parameter	Description
Type	The type of Service. This parameter determines how the Service is accessed. ■ **Cluster IP**: The ClusterIP type Service. This type of Service exposes the Service by using an internal IP address of the cluster. If you select this type, the Service is accessible only within the cluster. This is the default type. **Note** The **Headless Service** check box is displayed only when you set Type to **Cluster IP**. ■ **Node Port**: The NodePort type Service. This type of Service is accessed by using the IP address and a static port of each node. A NodePort Service can be used to route requests to a ClusterIP Service. The ClusterIP Service is automatically created by the system. You can access a NodePort Service from outside the cluster by sending requests to `<NodeIP>:<NodePort>`. ■ **Server Load Balancer**: The LoadBalancer type Service. This type of Service exposes the Service by using an SLB instance. If you select this type, you can enable internal or external access to the Service. SLB instances can be used to route requests to NodePort and ClusterIP Services. ■ Create SLB Instance: You can click **Modify** to change the specification of the SLB instance. ■ Use Existing SLB Instance: You can select an existing SLB instance. **Note** You can create an SLB instance or use an existing SLB instance. You can also associate an SLB instance with more than one Service. However, you must take note of the following limits: ■ If you use an existing SLB instance, the listeners of the SLB instance overwrite the listeners of the Service. ■ If a Classic Load Balancer (CLB) instance is created along with a Service, you cannot reuse this CLB instance when you create other Services. Otherwise, the CLB instance may be deleted. Only CLB instances that are manually created in the console or by calling the API can be used to expose multiple Services. ■ Kubernetes services that share the same CLB instance must use different frontend listening ports. Otherwise, port conflicts may occur. ■ If multiple Kubernetes Services share the same SLB instance, you must use the listener names and the vServer group names as unique identifiers in Kubernetes. Do not modify the names of listeners or vServer groups. ■ You cannot share SLB instances across clusters.

Parameter	Description
Port Mapping	Specify a Service port and a container port. The container port must be the same as the one that is exposed in the backend pod.
External Traffic Policy	■ Local: Traffic is routed to only the node where the Service is deployed. ■ Cluster: Traffic can be routed to pods on other nodes. **Note** The **External Traffic Policy** parameter is available only if you set Type to **Node Port** or **Server Load Balancer**.
Annotations	Add one or more annotations to the Service to configure the SLB instance. For example, `service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth:20` specifies that the maximum bandwidth of the Service is 20 Mbit/s. This limits the amount of traffic that flows through the Service.
Label	Add one or more labels to the Service. Labels are used to identify the Service.

- To create an Ingress, click **Create** to the right side of **Ingresses**. In the Create dialog box, set the parameters.

  For more information about the parameters that are required for creating an Ingress, see Ingress configurations.

  Notice When you create an application from an image, you can create an Ingress only for one Service. In this example, a virtual hostname is specified as the test domain name. You must add a mapping to the *hosts* file for this domain name in the following format: *External endpoint of the Ingress* + *domain name of the Ingress*. In actual scenarios, use a domain name that has obtained an Internet Content Provider (ICP) number.

  ```
  101.37.xx.xx    foo.bar.com    # The IP address of the Ingress.
  ```

Parameter	Description
Name	The name of the Ingress.

Parameter	Description
Rules	Ingress rules are used to enable access to specified Services in a cluster. For more information, see Ingress configurations. ■ **Domain**: Enter the domain name of the Ingress. ■ **Path**: Enter the Service URL. The default path is the root path /. The default path is used in this example. Each path is associated with a backend Service. SLB forwards traffic to a backend Service only when inbound requests match the domain name and path. ■ **Services**: Select a Service and a Service port. ■ **EnableTLS**: Select this check box to enable TLS.
Weight	Set the weight for each Service in the path. Each weight is calculated as a percentage value. Default value: 100.
Canary Release	After a canary release rule is configured, only requests that match the rule are routed to the Service of the new application version. If the weight of new-nginx is lower than 100%, requests that match the specified rules are routed to the Service based on the Service weight. Container Service supports multiple traffic splitting methods. This allows you to select suitable solutions for specific scenarios, such as canary releases and A/B testing: ■ Traffic splitting based on request headers ■ Traffic splitting based on cookies ■ Traffic splitting based on query parameters **Note** Only Ingress controllers of 0.12.0-5 and later versions support traffic splitting. The following parameters are required: ■ **Services**: Specify the Services to be accessed. ■ **Type**: Select the type of matching rule. Valid values: Header, Cookie, and Query. ■ **Name** and **Match Value**: user-defined request parameters that are specified in key-value pairs. ■ **Matching Rule**: Regular expressions and exact matches are supported.

Parameter	Description
Annotations	Click the + icon to add an annotation. You can select **Custom Annotation** or **Ingress-NGINX** from the Type drop-down list. For more information about Ingress annotations, see Annotations. **Note** If you want to add an annotation for configuring rewrite rules, select **Ingress-NGINX** from the Type drop-down list. Select **nginx.ingress.kubernetes.io/rewrite-target Rewrite Annotation** from the **Name** drop-down list. Then, specify / in the **Value** column. This annotation specifies that the requests destined for */path* are redirected to the root path, which can be recognized by the backend Service.
Labels	Add labels to describe the characteristics of the Ingress.

You can find the created Service and Ingress in the **Access Control** section. You can click **Update** or **Delete** to change the configurations.

2. (Optional)In the **Scaling** section, specify whether to enable **HPA**. Horizontal Pad Autoscaler (HPA) allows you to meet the resource requirements of the application at different load levels.

   HPA can automatically scale the number of pods in a Container Service cluster based on the CPU and memory usage.

   Note To enable HPA, you must configure resources required by the container. Otherwise, HPA does not take effect.

Parameter	Description
Metric	Select CPU Usage or Memory Usage. The selected resource type must be the same as that specified in the Required Resources field.
Conditions	Specify the resource usage threshold. HPA triggers scale-out events when the threshold is exceeded.
Max. Replica	Specify the maximum number of pods to which the application can be scaled.
Min. Replica	Specify the minimum number of pods that must run.

3. (Optional)In the **Scheduling** section, you can configure the following parameters: **Update**

Method, **Node Affinity**, **Pod Affinity**, **Pod Anti Affinity**, and **Toleration**. For more information, see Affinity and anti-affinity.

> **Note** Node affinity and pod affinity affect pod scheduling based on node labels and pod labels. You can add node labels and pod labels that are provided by Kubernetes to configure node affinity and pod affinity. You can also add custom labels to nodes and pods, and then configure node affinity and pod affinity based on these custom labels.

Parameter	Description
Update Method	Select Rolling Update or OnDelete. For more information, see Deployments.
Node Affinity	Set **Node Affinity** by selecting worker node labels as selectors. Node affinity supports required and preferred rules, and various operators such as In, NotIn, Exists, DoesNotExist, Gt, and Lt. ◦ **Required**: Specify the rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution affinity. These rules have the same effect as the `NodeSelector` parameter. In this example, pods can be scheduled only to nodes with the specified labels. You can create multiple required rules. However, only one of them must be met. ◦ **Preferred**: Specify the rules that are not required to be matched for pod scheduling. In the YAML file, these rules are defined by the preferredDuringSchedulingIgnoredDuringExecution affinity. If you specify a preferred rule, the scheduler attempts to schedule a pod to a node that matches the preferred rule. You can also set weights for preferred rules. If multiple nodes match the rule, the node with the highest weight is preferred. You can create multiple preferred rules. However, all of them must be met before the pod can be scheduled.

Parameter	Description
Pod Affinity	Pod affinity rules specify how pods are deployed relative to other pods in the same topology domain. For example, you can use pod affinity to deploy services that communicate with each other to the same topological domain, such as a host. This reduces the network latency between these services. Pod affinity enables you to specify to which node pods can be scheduled based on the labels of running pods. Pod affinity supports required and preferred rules, and the following operators: `In, NotIn, Exists, and DoesNotExist`. ○ **Required**: Specify rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution affinity. A node must match the required rules before pods can be scheduled to the node. ■ **Namespace**: Specify the namespace to apply the required rule. Pod affinity rules are defined based on the labels that are added to pods and therefore must be scoped to a namespace. ■ **Topological Domain**: Set the topologyKey. This specifies the key for the node label that the system uses to denote the topological domain. For example, if you set the parameter to `kubernetes.io/hostname`, topologies are determined by nodes. If you set the parameter to `beta.kubernetes.io/os`, topologies are determined by the operating systems of nodes. ■ **Selector**: Click the plus icon to the right of Selector. You can specify multiple selectors of the rule. You can set the Label Name, Operator, Label Value parameters for each selector. In this example, the required rule specifies that the application to be created is scheduled to a host that runs applications with the `app:nginx` label. ■ **View Applications**: Click **View Applications** and set the namespace and application in the dialog box that appears. You can view the pod labels of the selected application and add these labels as selectors. ○ **Preferred**: Specify the rules that are not required to be matched for pod scheduling. In the YAML file, these rules are defined by the preferredDuringSchedulingIgnoredDuringExecution affinity. If you specify a preferred rule, the scheduler attempts to schedule the pod to a node that matches the preferred rule. You can set weights for preferred rules. The other parameters are the same as those of required rules. **Note** **Weight**: Set the weight of a preferred rule to a value from 1 to 100. The scheduler calculates the weight of each node that meets the preferred rule based on an algorithm, and then schedules the pod to the node with the highest weight.

Parameter	Description
Pod Anti Affinity	Pod anti-affinity rules specify that pods are not scheduled to topological domains where pods with matching labels are deployed. Pod anti-affinity rules apply to the following scenarios: ○ Schedule the pods of an application to different topological domains, such as multiple hosts. This allows you to enhance the stability of your service. ○ Grant a pod exclusive access to a node. This enables resource isolation and ensures that no other pods can share the resources of the specified node. ○ Schedule pods of an application to different hosts if the pods may interfere with each other. **Note** The parameters of pod anti-affinity rules are the same as those of pod affinity rules. Create rules based on scenarios.
Toleration	Configure toleration rules to allow pods to be scheduled to nodes with matching taints.

4. (Optional)In the **Labels,Annotations** section, click **Add** to configure labels and annotations for the pod.

Parameter	Description
Pod Labels	Add a label to the pod. The label is used to identify the application.
Pod Annotations	Add an annotation to the pod configurations.

5. Click **Create**.

Step 4: Check the application

1. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments** . On the Deployments page, you can find that the application that you created is displayed.
2. In the left-side navigation pane of the cluster details page, choose **Network > Ingresses** . On the Ingresses page, you can find that the Ingress that you created is displayed.
3. Enter the test domain name in the address bar of your browser and press Enter. The NGINX welcome page appears.

What to do next

View application details

In the left-side navigation pane of the Container Service console, click **Clusters**. Click the name of the cluster where the application is deployed or click **Details** in the **Actions** column. Choose **Workloads > Deployments**. On the **Deployments** page, click the name of the deployed application or click **Details** in the **Actions** column.

On the details page of the application, you can view the YAML file of the application. You can also edit, scale, redeploy, and refresh the application.

Operation	Description
Edit	On the details page of the application, click **Edit** in the upper-right corner of the page to modify the application configurations.
Scale	On the details page of the application, click **Scale** in the upper-right corner of the page to scale the application to a required number of pods.
View in YAML	On the details page of the application, click **View in YAML** to **update** or **download** the YAML file. You can also click **Save As** to save the file as a different name.
Redeploy	On the details page of the application, click **Redeploy** in the upper-right corner of the page to redeploy the application.
Refresh	On the details page of the application, click **Refresh** in the upper-right corner of the page to refresh the details page.

On the details page of the application, you can view information about the pods, pod events, historical versions of the application, application log, access methods of the application, and application triggers.

Item	Description
Pods	Click the **Pods** tab to view information about the pods that are provisioned for the application.
Access methods	Click the **Access Methods** tab to view information about the access methods of the application. You can also perform the following operations: • Click **Create** to the right side of **Services** to create a Service. For more information about the parameters required for creating a Service, see Create a service. • Click **Create** to the right side of **Ingresses** to create an Ingress. For more information about the parameters required for creating an Ingress, see Ingress configurations.
Pod events	Click the **Events** tab to view information about the pod events.

Item	Description
Pod scaling	Click the **Pod Scaling** tab to configure HPA to scale the application based on the CPU utilization and memory utilization of the application. Click **Create** to the right side of **HPA** to configure HPA. For more information about the parameters required for configuring HPA, see Metric scaling.
Historical application versions	Click the **History Versions** tab to view the historical versions of the application.
Application log	Click the **Logs** tab to view the application log.
Triggers	On the details page of the application, click the **Triggers** tab to create application triggers. For more information about application triggers, see Create a trigger on an application.

Manage the application

On the **Deployments** page, select the application and click **More** in the **Actions** column to perform the following operations.

Operation	Description
View in YAML	View the YAML file of the application.
Redeploy	Redeploy the application.
Edit Label	Configure the labels of the application.
Edit Annotations	Configure the annotations of the application.
Node Affinity	Configure the node affinity settings of the application. For more information, see Scheduling.
Scaling	Configure the scaling settings of the application. For more information, see Horizontal pod autoscaling.
Toleration	Configure the toleration rules of the application. For more information, see Scheduling.
Upgrade Policy	Configure the update policy of the application. • **Rolling Update**: Pods are updated in a rolling update fashion. • **OnDelete**: All existing pods are deleted before new pods are created.
Clone	Create a new application by using the same container settings as the current application.
Roll Back	Roll back the application to a previous version.
Logs	View the log of the application.

Operation	Description
Delete	Delete the application.

Create a Linux application by using an orchestration template

In an orchestration template, you must define the resource objects that are required for running an application and configure mechanisms such as label selectors to manage the resource objects that make up the application.

This section describes how to use an orchestration template to create an NGINX application that consists of a Deployment and a Service. The Deployment provisions pods for the application and the Service manages access to the backend pods.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**.
5. In the upper part of the **Deployments** page, select a namespace and click **Create from Template**.
6. Set the parameters and click **Create**.
   - **Sample Template**: Container Service provides YAML templates of various resource types. This simplifies the deployment of resource objects. You can also create a custom template based on the YAML syntax to define the resources that you want to create.
   - **Add Deployment**: This feature allows you to define a YAML template.
   - **Use Existing Template**: You can import an existing template to the configuration page.
   - **Save As**: You can save the template that you have configured.

   The following sample template is based on an orchestration template provided by Container Service. You can use this template to create a Deployment to run an NGINX application.
   - Container Service supports the YAML syntax. You can use the `---` symbol to separate multiple resource objects. This allows you to create multiple resource objects in a single template.
   - (Optional)By default, if you mount a volume to the application, the existing files in the mount path of the application are overwritten. To avoid the existing files from being overwritten, you can add a `subPath` parameter.

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
    name: nginx-deployment
    labels:
      app: nginx
spec:
    replicas: 2
    selector:
      matchLabels:
        app: nginx
    template:
      metadata:
        labels:
          app: nginx
      spec:
        containers:
        - name: nginx
          image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
          ports:
          - containerPort: 80
---
apiVersion: v1     # for versions before 1.8.0 use apps/v1beta1
kind: Service
metadata:
   name: my-service1        #TODO: to specify your service name
   labels:
     app: nginx
spec:
   selector:
     app: nginx             #TODO: change label selector to match your backend pod
   ports:
   - protocol: TCP
     name: http
     port: 30080            #TODO: choose an unique port on each node to avoid port con
flict
     targetPort: 80
   type: LoadBalancer        ##In this example, the type is changed from NodePort to Lo
adBalancer.
```

7. Click **Create**. A message that indicates the deployment status appears.

Manage applications by using commands

This topic describes how to create an application or view containers of an application by using commands.

Create an application by using commands

1. In a CLI, run the following command to start a container. An NGINX web server is used in this example.

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

2. Run the following command to create a service entry for this container. The `--type=LoadBalancer`

parameter in the command indicates that the system must create a Server Load Balancer (SLB) instance for the NGINX container.

```
kubectl expose deployment nginx --port=80 --target-port=80 --type=LoadBalancer
```

View containers by using commands

In the CLI, run the `kubectl get pods` command to list all running containers in the default namespace.

```
NAME                                READY     STATUS    RESTARTS   AGE
nginx-2721357637-d****              1/1       Running   1          9h
```

Create an application by using an image pull Secret

Container Service allows you to use image pull Secrets in the Container Service console. You can create an image pull Secret or use an existing image pull Secret.

When you create an application from a private image, you must set a Secret for image pulling to ensure the security of the image. In the Container Service console, you can specify the authentication information of the private image repository as a Secret of the docker-registry type. This Secret applies to the specified Kubernetes cluster.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, select the namespace and click **Create from Image** in the upper-right corner of the page.
6. On the **Basic Information** wizard page, set the parameters. For more information, see Configure basic settings.
7. Configure containers.

   The following example describes how to configure an image pull Secret. For more information about container configurations, see Basic container configurations.

   i. On the **Container** wizard page, enter a private image registry in the **Image Name** field. The private image registry must be in the following format: `domainname/namespace/imagename`.

   > **Note** You do not need to configure a image pulling Secret for a public image registry.

   ii. Enter the image version that you want to use in the Image Version field.

iii. Click **Set Image Pull Secret**, and create a Secret or select an existing Secret in the dialog box that appears.

- You can select **New Secret**. If you select New Secret, set the following parameters:

Parameter	Description
Name	The key name of the Secret. You can enter a custom name.
Repository Domain	The address of the specified Docker repository. If you use an image repository in Container Registry, the address is automatically specified.
Username	The username that is used to log on to the Docker repository. If you use an image repository in Container Registry, the username is the username of your Alibaba Cloud account.
Password	The password used to log on to the Docker repository. If you use an image repository in Container Registry, the password is the logon password of the image repository in Container Registry.
Email	The email address. This parameter is optional.

Click **OK**. The Secret appears on the page after it is created.

- You can also select **Existing Secret**. You can use commands or YAML files to create image pulling Secrets.

8. Configure other parameters based on your requirements. Then, click **Create**.

   For more information, see Step 3: Configure advanced settings.

9. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**. On the **Deployments** page, check the status of the application. If the application is running as normal, it indicates that the image is successfully pulled by using the Secret that you configured.

6.7.2. Use a StatefulSet to create a stateful application

Container Service allows you to deploy stateful applications by using StatefulSets in the Container Service console. This topic describes how to create a stateful NGINX application and the features of StatefulSets.

Prerequisites

- A Container Service cluster is created. For more information, see Create a Kubernetes cluster.
- A kubectl client is connected to the cluster. For more information, see Connect to a cluster through kubectl.
- A persistent volume claim (PVC) is created. For more information, see Create a PVC.

Background information

StatefulSets provide the following features.

Scenario	Description
Pod consistency	Pod consistency ensures that pods are started and terminated in the specified order and ensures network consistency. Pod consistency is determined by pod configurations, regardless of the node to which a pod is scheduled.
Stable and persistent storage	VolumeClaimTemplate allows you to mount a persistent volume (PV) to each pod. The mounted PVs are not deleted after you delete replicated pods or scale in the number of replicated pods.
Stable network identifiers	Each pod in a StatefulSet derives its `hostname` from the name of the StatefulSet and the ordinal of the pod. The pattern of the hostname is `StatefulSet name-pod ordinal`.
Stable orders	For a StatefulSet with N replicated pods, each pod is assigned an integer ordinal from 0 to N-1. The ordinals assigned to pods within the StatefulSet are unique.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > StatefulSets**.
5. In the top navigation bar of the **StatefulSets** page, select the namespace and click **Create from Image**.
6. On the **Basic Information** wizard page, configure the basic settings.

   In this example, the Type parameter is set to **StatefulSet** to deploy a stateful application.

Parameter	Description
Name	Enter a name for the application.
Replicas	The number of pods that are provisioned for the application.
Type	The type of the application. You can select **Deployment**, **StatefulSet**, **Job**, **CronJob**, or **DaemonSet**. In this example, **StatefulSet** is selected.
Label	Add labels to the application. The labels are used to identify the application.
Annotations	Add an annotation to the application.
Synchronize Timezone	Specify whether to synchronize the time zone between nodes and containers.

7. Click **Next** to proceed to the **Container** wizard page.

8. Configure containers.

> **Note** In the upper part of the **Container** wizard page, click **Add Container** to add more containers for the application.

The following table describes the parameters that are used to configure the containers.

- In the **General** section, configure the basic settings of the container.

Parameter	Description
Image	To select an image, click **Select Image**. In the upper right corner of the Select Image dialog box, select **Default** or **enterprise-edition** from the **Container Registry Instance** drop-down list. Then, select the image that you want to use and click **OK**. In this example, the nginx image is selected. **Note** ■ Default: shows images that are stored on the Container Registry Standard Edition instance. ■ enterprise-edition: shows images that are stored on the Container Registry Advanced Edition instance. You can also enter the address of a private image registry. Specify a private registry in the following format: `domainname/namespace/imagename`.
Image Version	■ Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used. ■ You can select the following image pulling policies: ■ **ifNotPresent**: If the image that you want to pull is found on your on-premises machine, the image on your on-premises machine is used. Otherwise, Container Service pulls the image from the image registry. ■ **Always**: Container Service pulls the image from the registry each time the application is deployed or expanded. ■ **Never**: Container Service uses only images on your on-premise machine. **Note** If you select **Image Pull Policy**, no image pulling policy is applied. ■ To pull the image without a password, click **Set Image Pull Secret** to configure a Secret for pulling images.
Resource Limit	You can specify an upper limit for the CPU, memory, and ephemeral storage space that the container can consume. This prevents the container from occupying an excessive amount of resources.

Parameter	Description
Required Resources	The amount of the CPU resources, memory resources, and ephemeral storage space that are reserved for this application. These resources are exclusive to the container. This prevents the application from becoming unavailable when other services or processes compete for computing resources.
Container Start Parameter	■ stdin: specifies that start parameters are sent to the container as standard input (stdin). ■ tty: specifies that start parameters defined in a virtual terminal are sent to the container. The two options are usually used together. In this case, the virtual terminal (tty) is associated to the stdin of the container. For example, an interactive program receives the stdin from the user and displays the content in the terminal.
Privileged Container	■ If you select Privileged Container, privileged=true is specified for the container and the privilege mode is enabled. ■ If you do not select Privileged Container, privileged=false is specified for the container and the privilege mode is disabled.
Init Container	If you select Init Container, an init container is created. An init container provides tools for pod management. For more information, see init containers.

- (Optional)In the **Ports** section, click **Add** to configure one or more container ports.
- (Optional)In the **Environments** section, click **Add** to set environment variables.

  You can configure environment variables in key-value pairs for pods. Environment variables are used to apply pod configurations to containers. For more information, see Pod variables.

Parameter	Description
Type	Select the type of environment variable. Valid values: ■ **Custom** ■ **ConfigMaps** ■ **Secret** ■ **Value/ValueFrom** ■ **ResourceFieldRef**
Variable	Specify the name of the environment variable.

Parameter	Description
Value/ValueFrom	Specify a reference that is used to define the environment variable.

- (Optional)In the **Health Check** section, you can enable **liveness** and **readiness** probes as needed.
- In the **Lifecycle** section, configure the lifecycle of the container.

  You can set the following parameters to configure the lifecycle of the container: Start, Post Start, and Pre Stop. For more information, see Attach Handlers to Container Lifecycle Events.
- (Optional)In the **Volume** section, you can mount local volumes and persistent volume claims (PVCs) to the container.
- (Optional)In the **Log** section, you can specify logging configurations and add custom tags to the collected log.

  Notice Make sure that the Log Service agent is installed in the cluster.

9. After you complete the basic configuration, click **Next**.
10. (Optional)Configure advanced settings.
    - In the **Access Control** section, you can configure access control settings for exposing backend pods.

      Note

      You can configure the following access control settings based on your business requirements:

      - Internal applications: For applications that run inside the cluster, you can create a ClusterIP or NodePort Service to enable internal communication.
      - External applications: For applications that are exposed to the Internet, you can configure access control by using one of the following methods:
        - Create a LoadBalancer Service and enable access to your application over the Internet by using a Server Load Balancer (SLB) instance.
        - Create an Ingress and use the Ingress to expose your application to the Internet. For more information, see Ingress.

      You can also specify how the backend pods are exposed to the Internet. In this example, a ClusterIP Service and an Ingress are created to expose the NGINX application to the Internet.

Parameter	Description

Parameter	Description
Service	Click **Create** to the right side of **Service**. In the **Create Service** dialog box, set the parameters. For more information about the parameters, see Create a service. **Cluster IP** is selected in this example.
Ingresses	Click **Create** to the right side of **Ingresses**. In the Create dialog box, configure the parameters. For more information about the parameters that are required to create an Ingress, see Create an Ingress in the console. **Note** When you create an application from an image, you can create an Ingress only for one Service. In this example, a virtual hostname is used as the test domain name. You must add the following entry to the hosts file to point the domain name to the IP address of the Ingress. In actual scenarios, use a domain name that has obtained an Internet Content Provider (ICP) number. `101.37.xx.xx foo.bar.com # The IP address of the Ingress.`

You can find the created Service and Ingress in the **Access Control** section. You can click **Update** or **Delete** to change the configurations.

- In the **Scaling** section, specify whether to enable **HPA**. Horizontal Pod Autoscaler (HPA) allows you to meet the resource requirements of the application at different load levels.

  HPA can automatically scale the number of pods in a Container Service cluster based on the CPU and memory usage.

  Note To enable HPA, you must configure resources required by the container. Otherwise, HPA does not take effect.

Parameter	Description
Metric	Select CPU Usage or Memory Usage. The selected resource type must be the same as that specified in the Required Resources field.
Condition	Specify the resource usage threshold. HPA triggers scale-out events when the threshold is exceeded.
Max. Replicas	Specify the maximum number of replicated pods to which the application can be scaled.
Min. Replicas	Specify the minimum number of replicated pods that must run.

- In the **Scheduling** section, you can configure the following parameters: **Update Method**,

Node Affinity, **Pod Affinity**, **Pod Anti Affinity**, and **Toleration**. For more information, see Affinity and anti-affinity.

> **Note** Node affinity and pod affinity affect pod scheduling based on node labels and pod labels. You can add node labels and pod labels that are provided by Kubernetes to configure node affinity and pod affinity. You can also add custom labels to nodes and pods, and then configure node affinity and pod affinity based on these custom labels.

Parameter	Description
Update Method	Select Rolling Update or OnDelete. For more information, see Deployments.
Node Affinity	Set **Node Affinity** by selecting worker node labels as selectors. Node affinity supports required and preferred rules, and various operators such as In, NotIn, Exists, DoesNotExist, Gt, and Lt. ■ **Required**: Specify the rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution affinity. These rules have the same effect as the `NodeSelector` parameter. In this example, pods can be scheduled only to nodes with the specified labels. You can create multiple required rules. However, only one of them must be met. ■ **Preferred**: Specify the rules that are not required to be matched for pod scheduling. In the YAML file, these rules are defined by the preferredDuringSchedulingIgnoredDuringExecution affinity. If you specify a preferred rule, the scheduler attempts to schedule a pod to a node that matches the preferred rule. You can also set weights for preferred rules. If multiple nodes match the rule, the node with the highest weight is preferred. You can create multiple preferred rules. However, all of them must be met before the pod can be scheduled.
	Pod affinity rules specify how pods are deployed relative to other pods in the same topology domain. For example, you can use pod affinity to deploy services that communicate with each other to the same topological domain, such as a host. This reduces the network latency between these services. Pod affinity enables you to specify to which node pods can be scheduled based on the labels of running pods. Pod affinity supports required and preferred rules, and the following operators: `In, NotIn, Exists, and DoesNotExist`.

Parameter	Description
Pod Affinity	■ **Required**: Specify rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution affinity. A node must match the required rules before pods can be scheduled to the node. ■ **Namespace**: Specify the namespace to apply the required rule. Pod affinity rules are defined based on the labels that are added to pods and therefore must be scoped to a namespace. ■ **Topological Domain**: Set the topologyKey. This specifies the key for the node label that the system uses to denote the topological domain. For example, if you set the parameter to `kubernetes.io/hostname`, topologies are determined by nodes. If you set the parameter to `beta.kubernetes.io/os`, topologies are determined by the operating systems of nodes. ■ **Selector**: Click the plus icon to the right of Selector. You can specify multiple selectors of the rule. You can set the Label Name, Operator, Label Value parameters for each selector. In this example, the required rule specifies that the application to be created is scheduled to a host that runs applications with the `app:nginx` label. ■ **View Applications**: Click **View Applications** and set the namespace and application in the dialog box that appears. You can view the pod labels of the selected application and add these labels as selectors. ■ **Preferred**: Specify the rules that are not required to be matched for pod scheduling. In the YAML file, these rules are defined by the preferredDuringSchedulingIgnoredDuringExecution affinity. If you specify a preferred rule, the scheduler attempts to schedule the pod to a node that matches the preferred rule. You can set weights for preferred rules. The other parameters are the same as those of required rules. **Note** **Weight**: Set the weight of a preferred rule to a value from 1 to 100. The scheduler calculates the weight of each node that meets the preferred rule based on an algorithm, and then schedules the pod to the node with the highest weight.

Parameter	Description
Pod Anti Affinity	Pod anti-affinity rules specify that pods are not scheduled to topological domains where pods with matching labels are deployed. Pod anti-affinity rules apply to the following scenarios: ■ Schedule the pods of an application to different topological domains, such as multiple hosts. This allows you to enhance the stability of your service. ■ Grant a pod exclusive access to a node. This enables resource isolation and ensures that no other pods can share the resources of the specified node. ■ Schedule pods of an application to different hosts if the pods may interfere with each other. **Note** The parameters of pod anti-affinity rules are the same as those of pod affinity rules. Create rules based on scenarios.
Toleration	Configure toleration rules to allow pods to be scheduled to nodes with matching taints.

- In the **Labels and Annotations** section, click **Add** to configure labels and annotations for the pod.

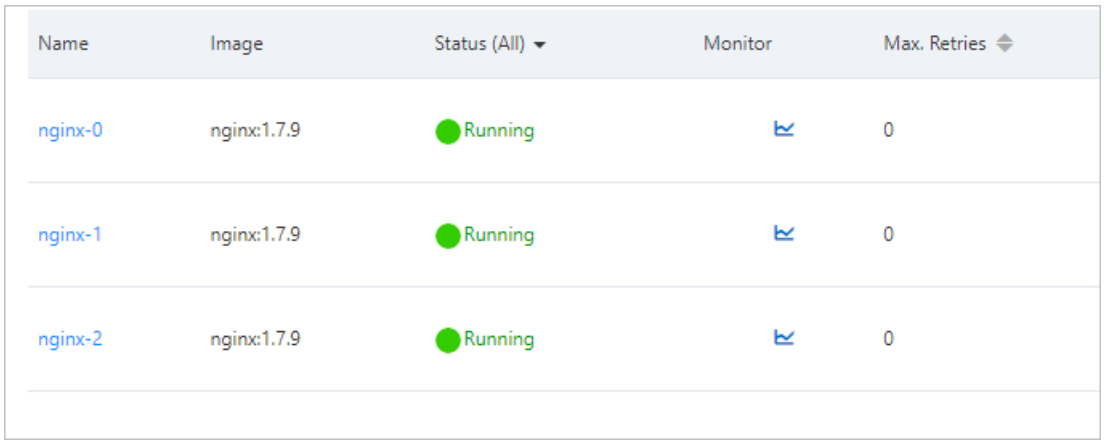
11. Click **Create**.

    In the left-side navigation pane, choose **Workloads > StatefulSets** to view the created stateful application.

12. (Optional)On the **StatefulSets** page, find the NGINX application and click **Scale** in the Actions column to scale the application.

    i. In the dialog box that appears, set Desired Number of Pods to 3.

    On the right side of the NGXIN application, click **Details** in the **Actions** column and then click the **Pods** tabs. After you scale out the application, all pods of the application are listed in an ascending order of ordinal indexes. If you scale in the application, pods are deleted in a descending order of ordinal indexes. This ensures that all pods follow a specific order.



ii. In the left-side navigation pane, choose **Volumes > Persistent Volume Claims**. Verify that after you scale out the application, new PVs and PVCs are created for the newly added pods. However, if the application is scaled in, existing PVs and PVCs are not deleted.

More operations

In the left-side navigation pane, click **Clusters**. Find the cluster that you want to manage and click **Applications** in the **Actions** column. In the left-side navigation pane of the cluster details page, choose **Workloads > StatefulSets**. On the **StatefulSets** page, click the name of the application or click **Details** in the **Actions** column.

> **Note** On the StatefulSets page, you can click **Label**, enter the key and `value` of a label added to the application, and then click **OK** to filter the applications.

On the details page of the application, you can **view the YAML file** of the application. You can also **edit**, **scale**, **redeploy**, and **refresh** the application.

- Edit: On the details page of the application, click **Edit** in the upper-right corner of the page to modify the configurations of the application.
- Scale: In the upper-right corner of the application details page, click **Scale** to scale the application to a required number of pods.
- View in YAML: On the details page of the application, click **View in YAML** in the upper-right corner of the page. In the Edit YAML dialog box, you can **update** and **download** the YAML file. You can also click **Save As** to save the *YAML* file as a different name.
- Redeploy: On the details page of the application, click **Redeploy** in the upper-right corner of the page to redeploy the application.
- Refresh: In the upper-right corner of the application details page, click **Refresh** to refresh the page.

What's next

Log on to a master node and run the following commands to test persistent storage.

1. Run the following command to create a temporary file in the disk:

   ```
   kubectl exec nginx-1 ls /tmp            # Query files (including lost+found) in the /tm
   p directory.
   kubectl exec nginx-1 touch /tmp/statefulset         # Create a file named statefulset.
   kubectl exec nginx-1 ls /tmp
   ```

   Expected output:

   ```
   lost+found
   statefulset
   ```

2. Run the following command to delete the pod to verify data persistence:

   ```
   kubectl delete pod nginx-1
   ```

   Expected output:

   ```
   pod"nginx-1" deleted
   ```

3. After the system recreates and starts a new pod, query the files in the /tmp directory. The following result shows that the statefulset file still exists. This shows the high availability of the

stateful application.

```
kubectl exec nginx-1 ls /tmp   # Query files (including lost+found) in the /tmp directo
ry to verify data persistence.
```

Expected output:

```
statefulset
```

6.7.3. Create a DaemonSet

A DaemonSet ensures that each node runs a copy of a pod. You can use a DaemonSet to run a log collection daemon, a monitoring daemon, or a system management application on each node. This topic describes how to create a DaemonSet for a Container Service cluster.

Create a DaemonSet in the Container Service console

Create a DaemonSet from an image

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > DaemonSets**.
5. On the **DaemonSets** page, click **Create from Image** in the upper-right corner.
6. Set parameters for the DaemonSet.
   i. On the **Basic Information** wizard page, configure the basic settings. For more information, see Create a stateless application by using a Deployment.
   ii. On the **Container** wizard page, configure one or more containers. For more information, see Create a stateless application by using a Deployment.
   iii. On the **Advanced** wizard page, configure the advanced settings.

   A DaemonSet can schedule a pod to a node that is in the Unschedulable state. To run a pod on only a specific node, set node affinity, pod affinity, or toleration rules. For more information about the configuration parameters, see Create a stateless application by using a Deployment.
7. Click **Create**.
   After the DaemonSet is created, you can view the DaemonSet on the DaemonSets page.

Create a DaemonSet from a YAML template

1. In the upper-right corner of the **DaemonSets** page, click **Create from YAML**.
2. On the **Create** page, configure the DaemonSet in the **Template** section.
3. Click **Create** below the **Template** section.
   After the DaemonSet is created, you can view the DaemonSet on the DaemonSets page.

Create a DaemonSet by using kubectl

Before you use kubectl to create a DaemonSet, you must download kubectl and connect to your cluster by using kubectl. For more information, see Connect to a cluster through kubectl.

A DaemonSet can schedule a pod to a node that is in the Unschedulable state. To run a pod on only a specific node, set the following parameters.

Parameter	Description
nodeSelector	A pod is scheduled only to the node with the specified labels.
nodeAffinity	Node affinity. Pods are scheduled to nodes based on node labels. Node affinity allows you to set other matching rules.
podAffinity	Pod affinity. Pods are scheduled to nodes based on pod labels. A pod is scheduled only to the node that runs a pod that matches the affinity rules.

To demonstrate how to create a DaemonSet by using kubectl, a DaemonSet named fluentd-elasticsearch is created in this example.

1. Create a *daemonset.yaml* file and copy the following content into the file:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      tolerations:
      # this toleration is to have the daemonset runnable on master nodes
      # remove it if your masters can't run pods
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: fluentd-elasticsearch
        image: quay.io/fluentd_elasticsearch/fluentd:v2.5.2
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
      terminationGracePeriodSeconds: 30
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
```

2. Run the following command to create the DaemonSet:

```
kubectl create -f daemonset.yaml
```

If `daemonset.apps/fluentd-elasticsearch created` is returned, the DaemonSet is created.

More operations

After you create a DaemonSet, you can perform the following operations:

- On the DaemonSets page, find the DaemonSet that you want to manage and click **Details** in the **Actions** column. On the details page, you can view basic information about the DaemonSet. The information includes pods, access method, events, and logs.
- On the DaemonSets page, find the DaemonSet that you want to manage and choose **More > View in YAML** in the **Actions** column to view the YAML file of the DaemonSet.
- On the DaemonSets page, find the DaemonSet that you want to manage and choose **More > Delete** in the **Actions** column to delete the DaemonSet.

6.7.4. Create a Job

You can use a Job to create an application in the Container Service console. This topic describes how to use a Job to create a busybox application and the operations that you can perform on the application.

Prerequisites

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

Context

A Job processes multiple short-lived, one-off tasks at a time to ensure that one or more pods in the tasks terminate with success.

Kubernetes supports the following types of Jobs:

- Non-parallel Jobs: In most cases, a Job of this type starts only one pod unless the pod fails. The Job is considered complete when the pod terminates with success.
- Jobs with a fixed completion count: A non-zero positive value is specified for `.spec.completions`. A Job of this type starts pods one after one. The Job is considered complete when the number of pods that terminate with success is equal to the value of `.spec.completions`.
- Parallel Jobs with a work queue: A non-zero positive value is specified for `.spec.Parallelism`. A Job of this type starts multiple pods at a time. The Job is considered complete when all pods terminate and at least one pod terminates with success. `.spec.completions` is not required.
- Parallel Jobs with a fixed completion count: A Job of this type has both `.spec.completions` and `.spec.Parallelism` specified. The Job starts multiple pods at a time to process a work queue.

Jobs can manage pods based on the settings of `.spec.completions` and `.spec.Parallelism` as described in the following table.

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

Job type	Example	Action	completions	Parallelism
One-off Job	Database migration	The Job starts only one pod. The Job is complete when the pod terminates with success.	1	1

Job type	Example	Action	completions	Parallelism
Job with a fixed completion count	Start pods one after one to process a work queue	The Job starts pods one after one. The Job is complete when the number of pods that terminate with success reaches the value of .spec.completions .	2+	1
Parallel Job with a fixed completion count	Start multiple pods at a time to process a work queue	The Job starts multiple pods at a time. The Job is complete when the number of pods that terminate with success reaches the value of .spec.completions .	2+	2+
Parallel Job	Start multiple pods at a time to process a work queue	The Job starts one or more pods at a time. The Job is complete when at least one pod terminates with success.	1	2+

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Jobs**.
5. On the **Jobs** page, click **Create from Image** in the upper-right corner.
6. On the **Basic Information** wizard page, configure the basic settings. For more information, see Create a stateless application by using a Deployment. Then, click **Next**.
7. On the **Container** wizard page, configure one or more containers. For more information, see Create a stateless application by using a Deployment. Then, click **Next**.
8. On the **Advanced** wizard page, configure the Job and add labels and annotations.
   - In the **Job Settings** section, set the following parameters.

Parameter	Description
Completions	The number of pods that must terminate with success. Default value: 1.
Parallelism	The number of pods that the Job must run in parallel. Default value: 1.
Timeout	The activeDeadlineSeconds field in the YAML file. This field specifies the duration time limit of the Job. If the Job is not complete within the duration time limit, the system attempts to terminate the Job.
BackoffLimit	The backoffLimit field in the YAML file. This field specifies the number of retries that are performed by the Job upon failure. Default value: 6. Failed pods are recreated with an exponential back-off delay. The upper limit of the delay is 6 minutes.
Restart	Valid values: Never and On Failure.

   - In the **Labels and Annotations** section, click **Add** to configure labels and annotations for the pod.

9. Click **Create**.

   In the left-side navigation pane, choose **Workloads > Jobs**. On the **Jobs** page, you can view the created Job. Check the status of the pods that are provisioned for the Job in the **Status** column. In this example, two pods are created to run in parallel based on the Job configuration.

   > **Note** If the Job is complete, you can view the end time on the **Jobs** page. If the Job is not complete, the end time is not displayed.

More operations

In the left-side navigation pane, click **Clusters**. Find the cluster that you want to manage and click the cluster name or click **Details** in the **Actions** column. On the details page of the cluster, choose **Workloads > Jobs**. Find the Job that you want to manage and click the Job name or click **Details** in the **Actions** column. On the Job details page, you can **scale** and **refresh** the application. You can also **view the YAML file** of the application.

- Scale: In the upper-right corner of the Job details page, click **Scale** to scale the application to a required number of pods.
- View the YAML file: In the upper-right corner of the Job details page, click **View in YAML**. Then, you can **update** and **download** the YAML file. You can also **save** the *YAML* file as a template.
- Refresh: In the upper-right corner of the Job details page, click **Refresh** to refresh the Job.

6.7.5. Create a CronJob

CronJobs are used to create periodic and recurring tasks. For example, you can create CronJobs to run backups or send emails. Jobs are used to process short-lived, one-off tasks. A CronJob creates one or more Jobs based on a specific schedule. This topic describes how to create a CronJob.

Create a CronJob in the Container Service console

Create a CronJob from an image

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, click the name of the cluster that you want to manage, or click **Details** in the **Actions** column of the cluster.
4. In the left-side navigation pane of the details page, choose **Workloads > CronJobs**.
5. On the **CronJobs** page, click **Create from Image** in the upper-right corner.
6. Set parameters for the CronJob.
   i. On the **Basic Information** wizard page, configure the basic settings. For more information, see Create a stateless application by using a Deployment.
   ii. On the **Container** wizard page, configure one or more containers. For more information, see Create a stateless application by using a Deployment.
   iii. On the **Advanced** wizard page, configure the advanced settings.

Item	Parameter	Description

CronJobs	Schedule	You can specify a schedule on a daily, weekly, or monthly basis. You can also specify a cron expression. A cron expression is a string of six or seven fields that are separated with five or six spaces. Each field describes an individual detail of the schedule. Specify the cron expression in the following format:
		`Seconds Minutes Hours DayofMonth Month DayofWeek Year`
		`Seconds Minutes Hours DayofMonth Month DayofWeek`
		For more information, see Cron Expressions.

	Concurrency Policy	You can select one of the following concurrency polices: ■ Allow: allows Jobs to run concurrently. Concurrent Jobs compete for cluster resources. ■ Forbid: disallows Jobs to run concurrently. If a Job is not complete within the schedule, the next Job is skipped. ■ Replace: If a Job is not complete within the schedule, the Job is skipped.

Item	Parameter	Description
	Job History	You can specify the numbers of successful or failed Jobs for which you want to retain records. If you set the parameters to 0, the system does not retain the records of Jobs.
Job Settings	Completions	For more information about how to set parameters in the Job Settings section, see Job settings.
	Parallelism	
	Timeout	
	BackoffLimit	
	Restart	
Labels and annotations	Pod Labels	You can add labels to pods in key-value pairs. **Note** The key of a label must be 1 to 253 characters in length, and can contain only letters, digits, hyphens (-), underscores (_), and periods (.).
	Pod Annotations	You can add annotations to pods in key-value pairs. **Note** The name of an annotation must be 1 to 253 characters in length, and can contain only letters, digits, hyphens (-), underscores (_), and periods (.).

7. Click **Create**.
   After the CronJob is created, you can view the CronJob on the CronJobs page.

Create a CronJob from a YAML template

1. In the upper part of the **CronJobs** page, select a namespace from the **Namespace** drop-down list and click **Create from Template** in the upper-right corner.
2. On the **Create** page, select a template and configure the template in the **Template** section.
3. Click **Create**.

Create a CronJob by using kubectl

Before you use kubectl to create a CronJob, you must download kubectl and connect to your cluster by using kubectl. For more information, see Connect to ACK clusters by using kubectl.

The following table describes the key parameters that are used to create a CronJob.

Parameter	Description
.spec.schedule	Specifies the schedule of the CronJob. For more information about the schedule format, see Cron schedule.

Parameter	Description
.spec.jobTemplate	Specifies the type of Job to be run. For more information about Job types, see Job patterns.
.spec.startingDeadlineSeconds	Specifies the due time before which a Job must be run.
.spec.concurrencyPolicy	Specifies the concurrency policy. Valid values: Allow, Forbid, and Replace. • Allow: allows Jobs to run concurrently. Concurrent Jobs compete for cluster resources. • Forbid: disallows Jobs to run concurrently. If a Job is not complete within the schedule, the next Job is skipped. • Replace: If a Job is not complete within the schedule, the Job is skipped.

A CronJob named hello is created in this example to demonstrate how to create a CronJob by using kubectl.

1. Create a *cronjob.yaml* file and copy the following content into the file:

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
```

2. Run the following command to create a CronJob:

```
kubectl create -f cronjob.yaml
```

If `cronjob.batch/hello created` is returned, the CronJob is created.

More operations

After you create a CronJob, you can perform the following operations:

- On the CronJobs page, find the created CronJob. Click **Details** in the **Actions** column to view basic information about the CronJob. The information includes the Job list, events, and logs.
- On the CronJobs page, find the created CronJob. You can choose **More > View in YAML** in the

Actions column to view the YAML file of the CronJob. You can also choose **More > Stop** to stop the CronJob or choose **More > Delete** to delete the CronJob.

6.7.6. Create a Service

You can create a Service for your application in the Container Service console to provide access to the application.

In Kubernetes, a Service is an abstraction that defines a logical set of pods and a policy by which to access the pods. This pattern is also known as a microservice. A label selector determines whether the set of pods can be accessed by the Service.

Each pod in Kubernetes clusters has its own IP address. However, pods are frequently created and deleted. Therefore, if you directly expose pods to external access, high availability is not ensured. Services decouple the frontend from the backend. The frontend clients do not need to be aware of which backend pods are used. This provides a loosely-coupled microservice architecture.

For more information, see Kubernetes Services.

Prerequisites

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

Step 1: Create a Deployment

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, select a namespace and click **Create from Template** in the upper-right corner.
6. Select a sample template or customize a template, and click **Create**.

   In this example, the template of an NGINX Deployment is used.

   ```
   apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1 kind: Deployment metad
   ata: name: nginx-deployment-basic labels: app: nginx spec: replicas: 2 selector: matchL
   abels: app: nginx template: metadata: labels: app: nginx spec: # nodeSelector: # env: t
   est-team containers: - name: nginx image: nginx:1.7.9 # replace it with your exactly <i
   mage_name:tags> ports: - containerPort: 80
   ```

   Query the state of the Deployment.

Step 2: Create a Service

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Network > Services**.
5. On the Services page, set **Namespace** and click **Create** in the upper-right corner.

6. In the Create Service dialog box, set the parameters.
   - **Name**: Enter a name for the Service. In this example, nginx-svc is used.
   - **Type**: Select the type of Service. This parameter specifies the Service is accessed. Valid values:
     - Cluster IP: a ClusterIP Service. This type of Service exposes pods through an internal IP address of the cluster. If you select this option, pods can be accessed from only within the cluster. This is the default value.
     - Node Port: a NodePort Service. This type of Service exposes pods by using the IP address and a static port of each node. A NodePort Service can be used to route requests to a ClusterIP Service, which is automatically created by the system. You can access a NodePort Service from outside the cluster by sending requests to `<NodeIP>:<NodePort>`.
     - Server Load Balancer: a LoadBalancer Service. This type of Service exposes pods by using Server Load Balancer (SLB) instances, which support Internet access or internal access. SLB instances can be used to route requests to NodePort and ClusterIP Services.
   - **Backend**: the backend object that you want to associate with the Service. In this example, select nginx-deployment-basic that was created in the preceding step. If you do not associate the Service with a backend object, no Endpoint object is created. You can also manually associate the Service with an Endpoint object. For more information, see Create a Service without selectors.
   - **Port Mapping**: Set the Service port and container port. The container port must be the same as the one that is exposed in the backend pod.
   - **Annotations**: Add one or more annotations to the Service to configure SLB parameters. For example, set the name to `service.beta.kubernetes.io` and the value to `20`. This means that the maximum bandwidth of the Service is 20 Mbit/s. For more information, see Access services by using SLB.
   - **Label**: Add one or more labels to the Service. The labels are used to identify the Service.
7. Click **Create**. After the nginx-svc Service is created, it appears on the Services page.

   On the **Services** page, you can view basic information about the Service. You can also access its external endpoint by using a browser.

6.7.7. View a Service

You can view details about a Service in the Container Service console.

Context

Each pod in Kubernetes clusters has its own IP address. However, pods are frequently created and deleted. Therefore, it is not practical to directly expose pods to external access. Services decouple the frontend from the backend, which provides a loosely-coupled microservice architecture.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Network > Services**.

5. On the **Services** page, select the namespace, find the Service that you want to view, and then click **Details** in the Actions column.
   On the details page, you can view detailed information about the Service.

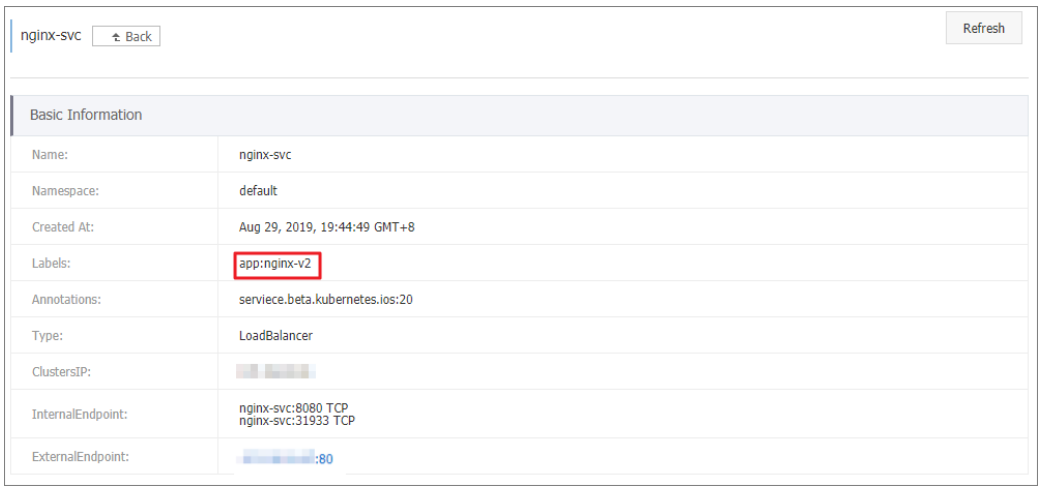
6.7.8. Update a Service

You can update a Service in the Container Service console. This topic describes how to update a Service.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Network > Services**.
5. On the **Services** page, select the namespace, find the Service that you want to update, and then click **Update** in the Actions column.
6. In the **Update Service** dialog box, modify the configurations and click **Update**.

   In the Service list, find the Service that you updated and click **Details** in the Actions column to view configuration changes. In this example, the labels of the Service are modified.



6.7.9. Delete a Service

This topic describes how to delete a Service.

Prerequisites

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

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions**

column.

4. In the left-side navigation pane of the details page, choose **Network > Services**.
5. On the Services page, select the namespace, find the Service that you want to delete, and then click **Delete** in the Actions column.
6. In the message that appears, click **Confirm**.

6.7.10. Use a trigger to redeploy an application

You can create a trigger and use it to redeploy an application. This topic describes how to use a trigger.

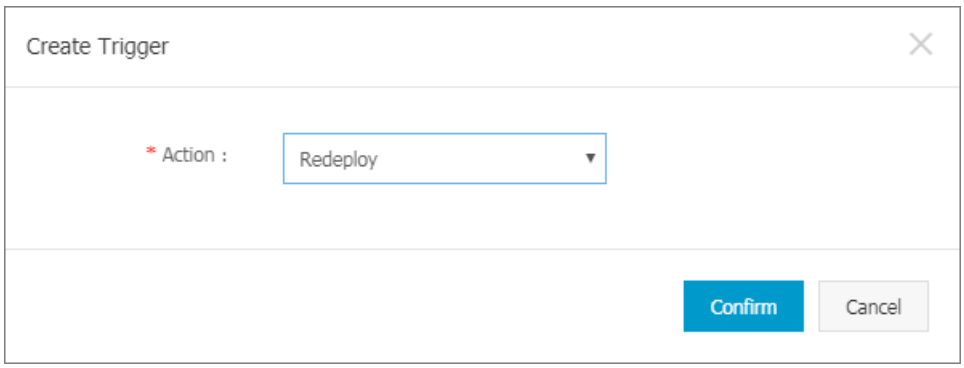
Prerequisites

- A Kubernetes cluster is created. For more information, see Create a Kubernetes cluster.
- An application is created. Then, a trigger is created for the application and the application is used to test the trigger. In this example, an NGINX application is created.

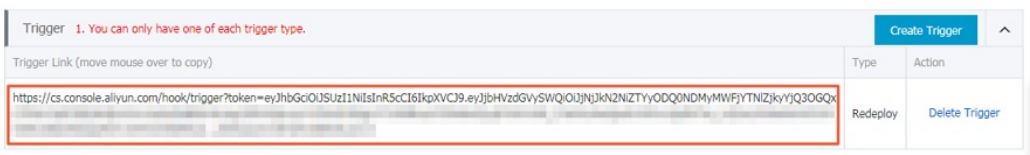
Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Manage** in the **Actions** column of the cluster.
4. In the left-side navigation pane, click Deployments. On the **Deployments** page, find the NGINX application and click **Details** in the Actions column.
5. On the details page of the application, click the **Triggers** tab. Then, click **Create Trigger**.
6. In the dialog box that appears, set Action to **Redeploy** and click **Confirm**.

   Note You can create triggers only to redeploy applications.

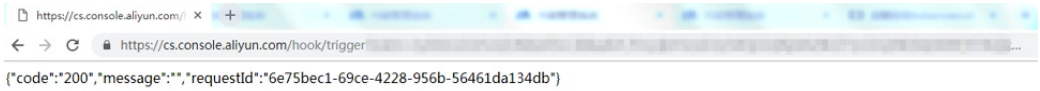


   After the trigger is created, the webhook URL of the trigger appears in the Trigger Link Address column on the **nginx - deployment** page.

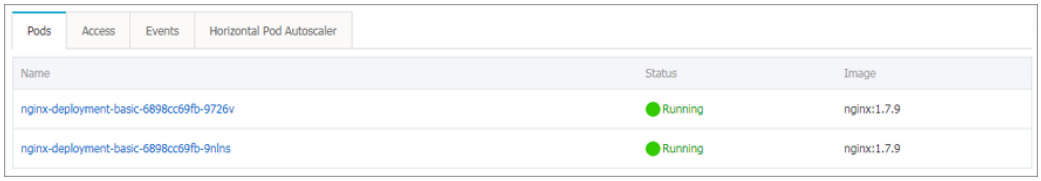


7. Copy the webhook URL, enter it into the address bar of your browser, and press Enter. A message

that indicates specific information such as the request ID appears.



8. Go to the **nginx - deployment** page. A new pod appears on the **Pods** tab.



After the new pod is deployed, the original pod is automatically deleted.

What's next

You can run triggers by sending GET or POST requests from an external system. For example, you can use the **curl** command-line tool to run triggers.

To run the redeploy trigger, run the following command:

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

6.7.11. Use kritis-validation-hook to automatically verify the signatures of container images

This topic describes how to use kritis-validation-hook to automatically verify the signatures of container images. This ensures that only images signed by trusted authorities are deployed and reduces the risk of malicious code execution.

Prerequisites

The container image is signed. For more information, see the *Sign container images* chapter in *Container Registry Standard Edition* of *Container Registry User Guide*.

Procedure

1. Install the image signature verification component.
   i. Log on to the Container Service console.
   ii. In the left-side navigation pane, click **Clusters**.
   iii. On the **Clusters** page, click the name of the cluster that you want to manage, or click **Details** in the **Actions** column of the cluster.
   iv. In the left-side navigation pane, choose **Operations > Add-ons**.
   v. In the **Optional Add-ons** section of the **Add-ons** page, find kritis-validation-hook and click **Install**.
2. Configure the signature verification policy.

i. Set the following signature parameters in the CLI of your on-premises machine to configure the signature verification policy:

```
export namespace=Actual value of namespace
export noteName=Actual value of noteName
export publicKeyData=Actual value of publicKeyData
```

- namespace: the namespace of Container Registry that is used for signing an image.
- noteName: the noteName setting used for signing an image.
- publicKeyData: the Base64-encoded GPG public key.

ii. Connect to a cluster through kubectl.

iii. Run the following command to set AttestationAuthority:

> **Note** The default namespace is used in the following example.

```
cat <<EOF > aa.yaml
apiVersion: kritis.grafeas.io/v1beta1
kind: AttestationAuthority
metadata:
  name: ${noteName}
spec:
  noteReference: namespaces/${namespace}
  publicKeyData: ${publicKeyData}
EOF
```

```
kubectl -n default apply -f aa.yaml
```

iv. Run the following command to set GenericAttestationPolicy:

> **Note** The default namespace is used in the following example.

```
cat <<EOF > gap.yaml
apiVersion: kritis.grafeas.io/v1beta1
kind: GenericAttestationPolicy
metadata:
  name: my-gap
spec:
  attestationAuthorityNames:
  - ${noteName}
EOF
```

```
kubectl -n default apply -f gap.yaml
```

3. Verify the signature verification feature.

In this example, the registry.acs.example.com/kritis-test/signed@sha256:2f122941b5850006dbb7adda78d2ea5b382841ca6569fd174bd24c14bfff**** image is signed. The registry.acs.example/kritis-test/not-sign@sha256:efc961b2b3499c25753d3c9f29977f494f49125cf1191071057aa68bffa7**** image is not signed. If the feature functions as expected, the signature verification policy enables the signed image and disables the unsigned image for the default namespace. If you use an unsigned image when you create an application, the application fails to be deployed and the following error message is returned: `admission webhook "kritis-validation-hook-deployments.grafeas.io" denied the request`. Run the following commands to verify the signature verification feature.

```
#A Deployment is created by using the signed image.
kubectl -n default run test-signed --image=registry.acs.example.com/kritis-test/signed@sha256:2f122941b5850006dbb7adda78d2ea5b382841ca6569fd174bd24c14bfff****
deployment.apps/test-signed created
#A Deployment fails to be created by using the unsigned image.
kubectl -n default run test-not-signed --image=registry.acs.example/kritis-test/not-sign@sha256:efc961b2b3499c25753d3c9f29977f494f49125cf1191071057aa68bffa7****
Error from server: admission webhook "kritis-validation-hook-deployments.grafeas.io" denied the request: image registry.acs.example/kritis-test/not-sign@sha256:efc961b2b3499c25753d3c9f29977f494f49125cf1191071057aa68bffa7**** is not attested
```

Notice When you create a resource after signature verification is enabled, you must specify the image by using the digest of the image, such as @sha256:<hash>.

6.7.12. View pods

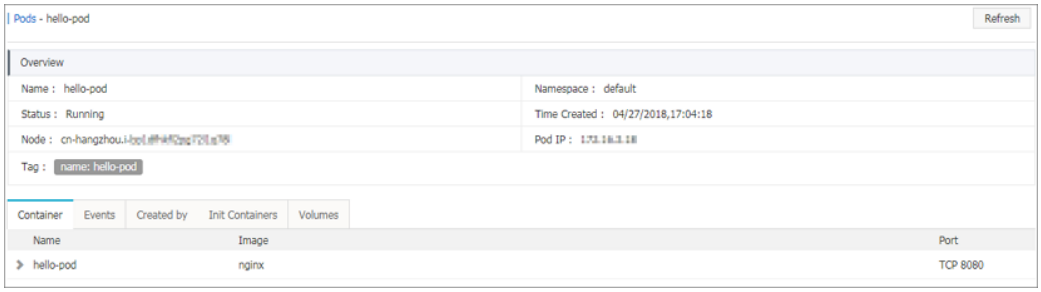
You can view pods in the Container Service console.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Pods**.
5. In the left-side navigation pane, choose **Applications > Pods**. The Pods page appears.
6. On the **Pods** page, select the namespace, find the pod that you want to view, and then click **View Details** in the Actions column.

   Note You can update or delete pods on the Pods page. We recommend that you use Deployments to manage pods if the pods are created by Deployments.

   On the details page of the pod, you can view detailed information about the pod.



6.7.13. Manage pods

Pods are the smallest deployable units in Kubernetes. A pod runs an instance of an independent application in Kubernetes. Each pod contains one or more containers that are tightly coupled. You can modify pods, view pods, and manually scale the number of pods for an application in the Container Service console.

View pods

View pod details

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Pods**.
5. Find the pod that you want to view and click **View Details** in the Actions column.

   You can view details of pods by using one of the following methods:

   - Method 1: In the left-side navigation pane of the details page, choose **Workloads > Deployments**. Find the application that you want to manage and click its name. On the **Pods** tab, click the name of the pod to view details.
   - Method 2: In the left-side navigation pane of the details page, choose **Network > Services**. Click the name of the Service that you want to manage. On the details page, find and click the name of the application that you want to manage. On the **Pods** tab, click the name of the pod to view details.

   > **Note** On the Pods page, you can modify and delete pods. For pods that are created by using a Deployment, we recommend that you use the Deployment to manage the pods.

View pod logs

You can view the logs of a pod by using one of the following methods:

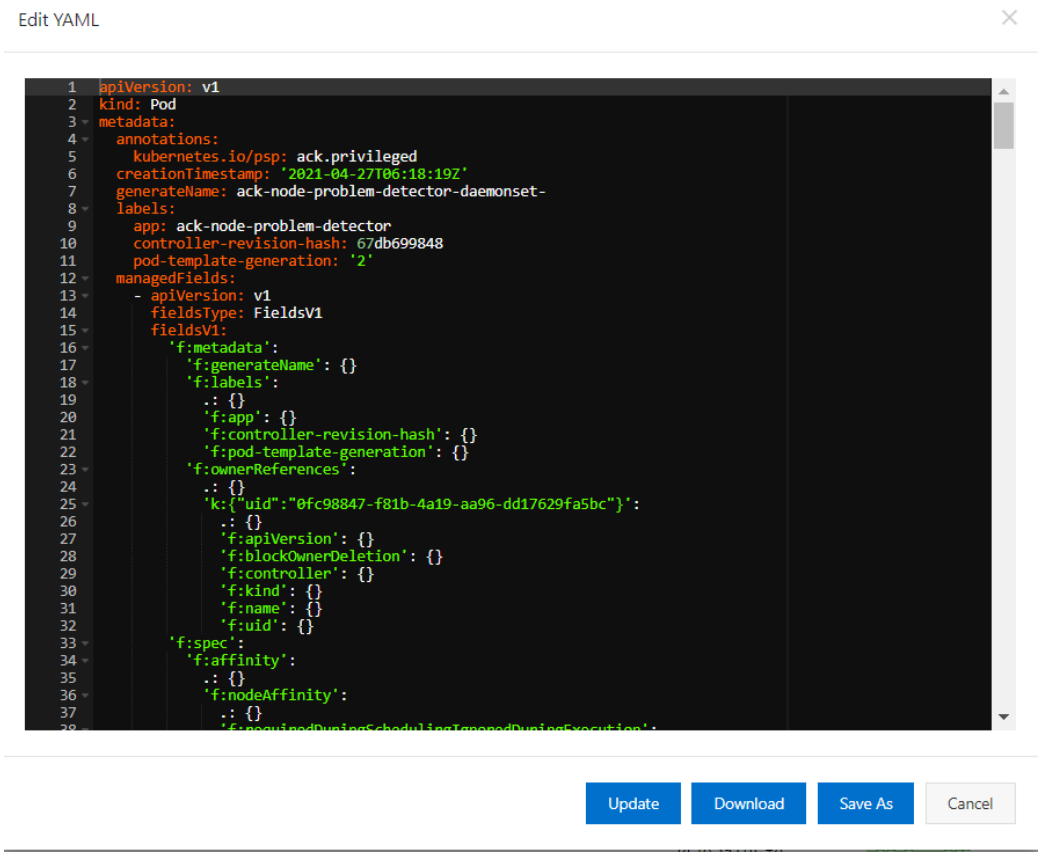
Navigate to the Pods tab, find the pod that you want to manage, and then click **Logs** on the right side of the page to view the log data.

Modify pod configurations

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details**

in the **Actions** column.

4. In the left-side navigation pane of the details page, choose **Workloads > Pods**.
5. On the **Pods** page, find the pod that you want to manage and click **Edit** on the right side of the page.
6. In the **Edit YAML** dialog box, modify the configurations based on your business requirements and click **Update**.



Manually scale pods for an application

After an application is created, you can scale the pods that are provisioned for the application.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. In the upper part of the **Deployments** page, select a **namespace**. Find the application that you want to manage and click **Scale** in the **Actions** column.
6. In the Scale dialog box, set Desired Number of Pods to 4 and click **OK**.

> **Note** By default, Deployments are updated based on the rollingUpdate strategy. This ensures that a minimum number of pods are available during the update. You can change this number in the YAML file.

Use a terminal to access a pod

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Pods**.
5. In the upper part of the **Pods** page, select a **namespace**. Find the pod that you want to manage and click **Terminal** in the **Actions** column.

   Then, you can run commands to manage the pod by using the terminal.

6.7.14. Schedule pods to specific nodes

You can add labels to nodes and schedule pods to the nodes with specified labels. This topic describes how to schedule a pod to a node with specified labels.

You can add labels to nodes and then configure `nodeSelector` to schedule pods to nodes with specified labels. For more information about how nodeSelector works, see nodeselector.

To meet business requirements, you may want to deploy a controller service on a master node, or deploy services on nodes that use standard SSDs. You can use the following method to schedule pods to nodes with specified labels.

Prerequisites

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

Step 1: Add a label to a node

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the **Nodes** page, click **Manage Labels and Taints** in the upper-right corner.
6. Select one or more nodes and then click **Add Label**. In this example, a worker node is selected.
7. In the dialog box that appears, enter the name and value of the label and click **OK**.

   On the Labels tab, you can find the `group:worker` label next to the selected node.

You can also run the following command to add a label to a node: `kubectl label nodes <node-name> <label-key>=<label-value>` .

Step 2: Schedule a pod to the node

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.

5. On the **Deployments** page, select the namespace and click **Create from Template** in the upper-right corner.
6. On the **Create** page, select a template from the Sample Template drop-down list. In this example, a custom template is selected. Copy the following content to the custom template and click **Create**.

   The following template is used as an example:

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    name: hello-pod
  name: hello-pod
spec:
  containers:
    - image: nginx
      imagePullPolicy: IfNotPresent
      name: hello-pod
      ports:
        - containerPort: 8080
          protocol: TCP
      resources: {}
      securityContext:
        capabilities: {}
        privileged: false
      terminationMessagePath: /dev/termination-log
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  nodeSelector:
    group: worker                          ## This value must be the same as the node
label that is added in Step 1.
 status: {}
```

6.7.15. Simplify application deployment by using Helm

This topic introduces the basic terms and components of Helm and describes how to use Helm to deploy an Apache Spark-based WordPress application in a Kubernetes cluster.

Prerequisites

- A Kubernetes cluster is created in the Container Service console. For more information, see Create a Kubernetes cluster.

  Tiller is automatically deployed to the cluster when the Kubernetes cluster is created. The Helm CLI is automatically installed on each master node. An Alibaba Cloud chart repository is added to Helm.

- A Kubernetes version that supports Helm is used.

  Only Kubernetes 1.8.4 and later support Helm. If the Kubernetes version of your cluster is 1.8.1, you can **upgrade** the cluster on the Clusters page of the Container Service console.

Context

Application management is the most challenging task in Kubernetes. The Helm project provides a unified method to package software and manage software versions. You can use Helm to simplify application distribution and deployment. App Catalog is integrated with Helm in the Container Service console and provides extended features based on Helm. App Catalog also supports Alibaba Cloud chart repositories to help you accelerate application deployments. You can deploy applications in the Container Service console or by using the Helm CLI.

This topic introduces the basic terms and components of Helm and describes how to use Helm to deploy an Apache Spark-based WordPress application in a Kubernetes cluster.

Basic terms

Helm is an open source project initiated by Deis. Helm can be used to simplify the deployment and management of Kubernetes applications.

Helm serves as a package manager for Kubernetes and allows you to find, share, and use applications built by Kubernetes. Before you use Helm, you must familiarize yourself with the following basic terms:

- Chart: a packaging format used by Helm. Each chart contains the images, dependencies, and resource definitions that are required to run an application. A chart may contain service definitions in a Kubernetes cluster. A Helm chart is similar to a Homebrew formula, an Advanced Package Tool (APT) dpkg, or a Yum rpm.
- Release: an instance of a chart that runs in a Kubernetes cluster. A chart can be installed multiple times in a Kubernetes cluster. After a chart is installed, a new release is created. For example, you can install a MySQL chart. If you want to run two databases in your cluster, you can install the MySQL chart twice. Each time a chart is installed, a release is created with a different name.
- Repository: the storage of charts. Charts are published and stored in repositories.

Helm components

Helm uses a client-server architecture and consists of the following components:

- The Helm CLI is the Helm client that runs on your on-premises machine or on the master nodes of a Kubernetes cluster.
- Tiller is the server-side component and runs in a Kubernetes cluster. Tiller manages the lifecycles of Kubernetes applications.
- A repository is used to store charts. The Helm client can access the index file and packaged charts in a chart repository over HTTP.

Deploy an application in the Container Service console

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the App Catalog page, find and click the WordPress chart to go to the details page of the chart.
4. Click the **Parameters** tab and modify the configurations.

   In this example, a dynamically provisioned disk volume is associated with a persistent volume claim (PVC). For more information, see Use Apsara Stack disks.

> **Note** You must first provision a disk as a persistent volume (PV). The capacity of the PV cannot be less than the capacity specified in the PVC.

5. In the **Deploy** section, select the cluster in which you want to deploy the application and click **Create**. After the application is deployed, you are redirected to the release page of the application.
6. In the left-side navigation pane, click **Clusters**.
7. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
8. In the left-side navigation pane of the details page, choose **Network > Services**. On the Services page, select the namespace, find the Service that is created for the application, and then check its HTTP and HTTPS external endpoints.
9. Click one of the endpoints to go to the WordPress website where you can publish blog posts.

Deploy an application by using the Helm CLI

After the Helm CLI is automatically installed on the master nodes of the Kubernetes cluster and the required chart repository is added to Helm, you can log on to the master nodes by using SSH. Then, you can deploy applications by using the Helm CLI. For more information, see Connect to a master node through SSH. You can also install and configure the Helm CLI and kubectl on your on-premises machine.

In this example, the Helm CLI and kubectl are installed and configured on your on-premises machine, and then an Apache Spark-based WordPress application is deployed.

1. Install and configure the Helm CLI and kubectl.
   i. Install and configure kubectl on your on-premises machine.

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

      To view the details of a Kubernetes cluster, run the `kubectl cluster-info` command.

   ii. Install Helm on your on-premises machine.

      For more information, see Install Helm.
2. Deploy the WordPress application.

   In the following example, a WordPress blog website is deployed by using Helm.

   i. Run the following command:

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

   > **Note** Container Service allows you to mount disks as dynamically provisioned volumes. Before you deploy the application, you must create a disk volume.

   The following output is returned:

   ```
   NAME: wordpress-testLAST DEPLOYED: Mon Nov 20 19:01:55 2017NAMESPACE: defaultSTATUS
   : DEPLOYED...
   ```

ii. Run the following command to query the release and Service that are created for the WordPress application:

```
helm listkubectl get svc
```

iii. Run the following command to view the pods that are provisioned for the WordPress application. You may need to wait until the pods change to the Running state.

```
kubectl get pod
```

iv. Run the following command to obtain the endpoint of the WordPress application:

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

You can enter the preceding URL in the address bar of your browser to access the WordPress application.

You can also run the following command based on the chart description to obtain the username and password of the administrator account for the WordPress application:

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

v. To delete the WordPress application, run the following command:

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

Use a third-party chart repository

You can use the default Alibaba Cloud chart repository. If a third-party chart repository is accessible from your cluster, you can also use the third-party chart repository. Run the following command to add a third-party chart repository to Helm:

```
helm repo add Repository name Repository URLhelm repo update
```

For more information about Helm commands, see Helm documentation.

References

Helm contributes to the development of Kubernetes. A growing number of software suppliers, such as Bitnami, have provided high-quality charts. For more information about available charts, visit `https://kubeapps.com/` .

6.8. SLB and Ingress

6.8.1. Overview

Container Service allows you to flexibly manage load balancing and customize load balancing policies for Kubernetes clusters. Kubernetes clusters provide you with a variety of methods to access containerized applications. They also allow you to use SLB or Ingress to access internal services and implement load balancing.

6.8.2. Use SLB to access Services

You can access a Service by using Server Load Balancer (SLB).

Procedure

1. Create an NGINX application by using a CLI.

```
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 a Service for the NGINX application and specify `type=LoadBalancer` to expose the Service to the Internet through an SLB instance.

```
root@master # kubectl expose deployment nginx --port=80 --target-port=80 --type=LoadBal
ancer
       root@master # kubectl get svc
       NAME                  CLUSTER-IP      EXTERNAL-IP      PORT(S)
AGE
       nginx                 172.19.XX.XX    101.37.XX.XX     80:31891/TCP
4s
```

3. Enter `http://101.37.XX.XX` in the address bar of your browser and press Enter to access the nginx Service.

SLB parameters

SLB provides a variety of parameters that you can use to configure features and services such as health check, billing method, and SLB instance type. For more information, see .

Annotations

You can add annotations to use the load balancing features provided by SLB.

Use an existing internal-facing SLB instance

Add two annotations. You must replace your-loadbalancer-id with the ID of your SLB instance.

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

Save the preceding code as an slb.svc file and run the `kubectl apply -f slb.svc` command.

Create an HTTPS-based LoadBalancer Service

You must first create a certificate in the SLB console. Then, you can use the certificate ID (cert-id) and the following template to create a LoadBalancer Service and an HTTPS-based SLB instance.

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

Note Annotations are case-sensitive.

SLB annotations

Annotation	Description	Default
service.beta.kubernetes.io/alicloud-loadbalancer-protocol-port	The listening port. Separate multiple ports with commas (,). Example: https:443,http:80.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-address-type	The type of the SLB instance. Valid values: internet and intranet.	internet
service.beta.kubernetes.io/alicloud-loadbalancer-slb-network-type	The network type of the SLB instance. Valid values: classic and vpc.	classic
service.beta.kubernetes.io/alicloud-loadbalancer-charge-type	The billing method of the SLB instance. Valid values: paybytraffic and paybybandwidth.	paybybandwidth
service.beta.kubernetes.io/alicloud-loadbalancer-id	The ID of an existing SLB instance. You can set the loadbalancer-id parameter to specify an existing SLB instance. In this case, the existing listeners are overwritten by the SLB instance. The SLB instance is not deleted when the Service is deleted.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-backend-label	The labels that are used to select the nodes to be added as the backend servers of the SLB instance.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-region	The region where the SLB instance is deployed.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-bandwidth	The bandwidth of the SLB instance.	50
service.beta.kubernetes.io/alicloud-loadbalancer-cert-id	The certificate ID. You must first upload the certificate.	""
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-flag	Valid values: on and off.	Default value: off. If TCP is used, do not modify this parameter. The health check feature is automatically enabled for TCP listeners and cannot be disabled.
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-type	For more information, see the *CreateLoadBalancerTCPListener* chapter of *SLB Developers Guide*.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-uri	For more information, see the *CreateLoadBalancerTCPListener* chapter of *SLB Developers Guide*.	N/A

Annotation	Description	Default
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-port	For more information, see the *CreateLoadBalancerTCPListener* chapter of *SLB Developers Guide*.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-healthy-threshold	For more information, see the *CreateLoadBalancerTCPListener* chapter of *SLB Developers Guide*.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-unhealthy-threshold	For more information, see the *CreateLoadBalancerTCPListener* chapter of *SLB Developers Guide*.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-interval	For more information, see the *CreateLoadBalancerTCPListener* chapter of *SLB Developers Guide*.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-connect-timeout	For more information, see the *CreateLoadBalancerTCPListener* chapter of *SLB Developers Guide*.	N/A
service.beta.kubernetes.io/alicloud-loadbalancer-health-check-timeout	For more information, see the *CreateLoadBalancerTCPListener* chapter of *SLB Developers Guide*.	N/A

6.8.3. Configure Ingress monitoring

You can enable the virtual host traffic status (VTS) dashboard to view Ingress monitoring data.

Enable the VTS dashboard by using the CLI

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

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

   The following template shows the modified Ingress ConfigMap:

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

2. Verify that the VTS dashboard is enabled for the NGINX Ingress controller.

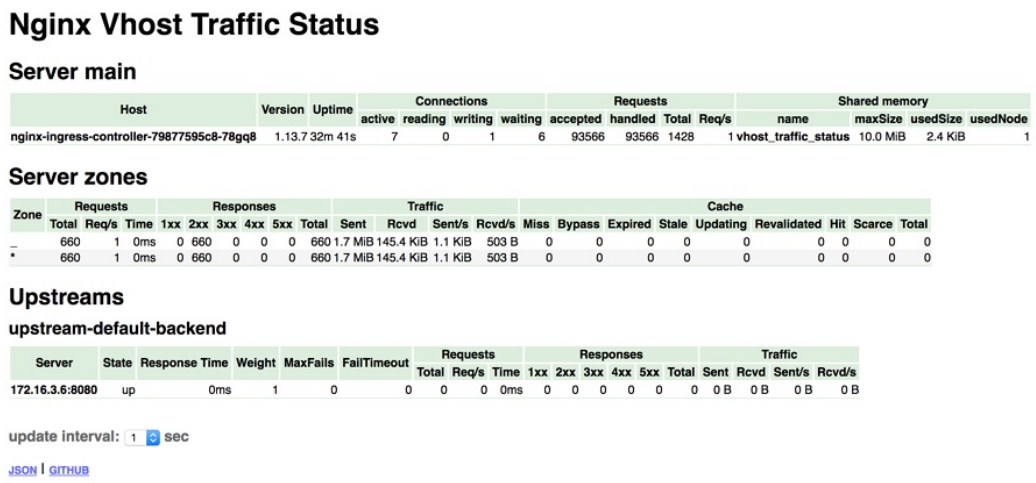
```
root@master # kubectl get pods --selector=app=ingress-nginx -n kube-system
NAME                                        READY     STATUS    RESTARTS   AGE
nginx-ingress-controller-79877595c8-78gq8   1/1       Running   0          1h
root@master # kubectl exec -it nginx-ingress-controller-79877595c8-78gq8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

3. Access the VTS dashboard from an on-premises machine.

> **Note** By default, the VTS port is not exposed due to security concerns. In the following example, port forwarding is used to access the VTS dashboard.

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

4. Visit `http://localhost:18080/nginx_status` to access the VTS dashboard.



Nginx Vhost Traffic Status

Server main

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

Server zones

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

Upstreams

upstream-default-backend

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

update interval: 1 sec

JSON | GITHUB

6.8.4. Ingresses

Kubernetes clusters support Ingress rules. You can define Ingress rules to meet your needs for load balancing.

In Kubernetes clusters, an Ingress is a set of routing rules that authorize external access to Services in clusters. You can use an Ingress to enable Layer 7 load balancing. You can configure an Ingress with URLs, Server Load Balancing (SLB) instances, SSL connections, and name-based virtual hosts to expose Services to external access.

Prerequisites

To test a complex routing scenario, an NGINX application is used in this example. A Deployment and multiple Services are created for the NGINX application. Replace Service names with the actual names.

```
kubectl run nginx --image=registry.cn-hangzhou.aliyuncs.com/acs/netdia:latest
kubectl expose deploy nginx --name=http-svc --port=80 --target-port=80
kubectl expose deploy nginx --name=http-svc1 --port=80 --target-port=80
kubectl expose deploy nginx --name=http-svc2 --port=80 --target-port80
kubectl expose deploy nginx --name=http-svc3 --port=80 --target-port=80
```

Create a simple Ingress

Run the following commands to create a simple Ingress that redirects traffic to the */svc* path to an http-svc Service. `nginx.ingress.kubernetes.io/rewrite-target: /` redirects traffic destined for the */svc* path to the */* path that can be recognized by the backend Service.

```
cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /svc
        backend:
          serviceName: http-svc
          servicePort: 80
EOF
```

```
kubectl get ing
NAME            HOSTS         ADDRESS           PORTS     AGE
simple          *             101.37.19*.***    80        11s
```

You can visit `http://101.37.19*.***/svc` to access the Service of the NGINX application.

Create a simple fanout Ingress that uses multiple domain names

You can create a simple fanout Ingress to route traffic to multiple Services with different domain names. The following example shows the configuration of a simple fanout Ingress:

```
cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple-fanout
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: http-svc1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: http-svc2
          servicePort: 80
  - host: foo.example.com
    http:
      paths:
      - path: /film
        backend:
          serviceName: http-svc3
          servicePort: 80
EOF
```

```
kubectl get ing
NAME            HOSTS         ADDRESS           PORTS     AGE
simple-fanout   *             101.37.19*.***    80        11s
```

After the preceding configuration is implemented, you can visit `http://foo.bar.com/foo` to access `http-svc1`, visit `http://foo.bar.com/bar` to access `http-svc2`, and visit `http://foo.example.com/film` to access `http-svc3`.

Note

- In a production environment, you must point the domain name to the returned address `101.37.192.211`.
- In a test environment, you must add the following mapping rules to the `hosts` file.

```
101.37.19*.*** foo.bar.com
101.37.19*.*** foo.example.com
```

Create a simple Ingress that uses the default domain name

If you have no domain names, you can create a simple ingress that uses the default domain name provided by Container Service. Then, you can use the default domain name to access Services. The default domain name is in the following format: `*.[cluster-id].[region-id].alicontainer.com`. You can find the default domain name in the basic information of the Kubernetes cluster in the Container Service console.

The following example shows the configuration of a simple Ingress that allows external access to Services through the default domain name:

```
cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: shared-dns
spec:
  rules:
  - host: foo.[cluster-id].[region-id].alicontainer.com  ## Replace with the default domain
name of your cluster.
    http:
      paths:
      - path: /
        backend:
          serviceName: http-svc1
          servicePort: 80
  - host: bar.[cluster-id].[region-id].alicontainer.com  ## Replace with the default domain
name of your cluster.
    http:
      paths:
      - path: /
        backend:
          serviceName: http-svc2
          servicePort: 80
EOF
```

```
kubectl get ing
NAME            HOSTS         ADDRESS          PORTS     AGE
shared-dns   foo.[cluster-id].[region-id].alicontainer.com,bar.[cluster-id].[region-id].ali
container.com             47.95.16*.***   80        40m
```

You can visit `http://foo.[cluster-id].[region-id].alicontainer.com/` to access Service `http-svc1` and visit `http://bar.[cluster-id].[region-id].alicontainer.com` to access Service `http-svc2`.

Create an Ingress to secure data transmission

Container Service allows you to use multiple types of certificates to reinforce the security of your applications.

1. Prepare a certificate.

   If you do not have a certificate, perform the following steps to generate a test certificate:

> **Note** The domain name must be the same as the one specified in your Ingress configurations.

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

After you run the preceding command, a certificate file *tls.crt* and a private key file *tls.key* are generated.

Use the certificate and private key to create a Kubernetes Secret named *foo.bar*. The Secret is referenced when you create the Ingress.

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

2. Create an Ingress to secure data transmission.

```
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
```

```
kubectl get ing
NAME            HOSTS         ADDRESS          PORTS     AGE
tls-fanout      *             101.37.19*.***   80        11s
```

3. You must configure the `hosts` file or set a domain name to access the tls Ingress, as described in the **Create a simple fanout Ingress that uses multiple domain names** section.

   You can visit `http://foo.bar.com/foo` to access `http-svc1` and visit `http://foo.bar.com/bar` to access `http-svc2`.

   You can also access the HTTPS Service by using HTTP. By default, an HTTPS Ingress redirects HTTP traffic to HTTPS. Therefore, access to `http://foo.bar.com/foo` is redirected to `https://foo.bar.com/foo`.

Create an Ingress

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Network > Ingresses**.
5. On the **Ingresses** page, select a namespace and click **Create Resources in YAML**.
6. On the **Create** page, select **Custom** from the **Sample Template** drop-down list, copy the following content into the template, and then click **Create**.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: simple
spec:
  rules:
  - http:
      paths:
      - path: /svc
        backend:
          serviceName: http-svc
          servicePort: 80
```

An Ingress that routes Layer 7 traffic to the `http-svc` Service is created.

6.8.5. Ingress configurations

Container Service provides Ingress controller components. Integrated with Apsara Server Load Balancer, these components provide Kubernetes clusters with flexible and reliable Ingress service.

An Ingress orchestration template is provided below. When you configure an Ingress through the console, you need to configure annotations and may need to create dependencies. For more information, see Create an ingress through the console, Ingress support, and Kubernetes Ingress. You can also create ConfigMaps to configure Ingresses. For more information, see https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/.

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

Annotations

For each Ingress, you can configure its annotations, Ingress controller, and rules, such as the route weight, canary release rule, and rewrite rules. For more information about annotations, see https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/.

For example, the following rewrite annotation, `nginx.ingress.kubernetes.io/rewrite-target: /`, indicates that */path* is redirected to the root path */*, which can be recognized by the backend service.

Rules

Ingress rules are used to manage external access to the services in the cluster and can be HTTP or HTTPS rules. You can configure the following items in rules: domain name (virtual hostname), URL path, service name, and port.

For each rule, you need to set the following parameters:

- Domain: The test domain or virtual hostname of your service, such as `foo.bar.com`.
- Path: The URL path of your service. Each path is associated with a backend service. Server Load Balancer only forwards traffic to the backend if the incoming request matches the domain and path.

- Service: Specify the service in the form of `service:port`. You also need to specify a route weight for each service. The Ingress routes traffic to the matching service based on the route weight.
  - Name: The name of the backend service.
  - Port: The port of the service.
  - Weight: The route weight of the service in the service group.

    > **Note**
    >
    > a. The weight is a percentage value. For example, you can set two services to the same weight of 50%.
    >
    > b. A service group includes services that have the same domain and path defined in the Ingress configuration. If no weight is set for a service, the default value, 100, is used.

Canary release

Container Service supports multiple traffic splitting approaches to suit scenarios such as canary release and A/B testing.

> **Note** Currently, only Ingress controllers of 0.12.0-5 and later versions support traffic splitting.

1. Traffic splitting based on request header
2. Traffic splitting based on cookie
3. Traffic splitting based on query parameter

After canary release is configured, only requests that match certain rules are routed to the corresponding service. If the weight of the corresponding service is lower than 100%, requests that match certain rules are routed to one of the services in the service group based on the weight.

TLS

You can use a Secret that contains a TLS private key and certificate to encrypt the Ingress. This ensures secure routing. The TLS Secret must contain a certificate named tls.crt and a private key named tls.key. For more information about how TLS works, see TLS. For how to create a Secret, see Configure a secure Ingress.

Labels

You can add labels to the Ingress.

6.8.6. Create an Ingress in the console

The Container Service console is integrated with the Ingress service. You can create an Ingress in the console and manage inbound traffic that is forwarded to different Services to meet your business requirements.

Prerequisites

- A Kubernetes cluster is created and an Ingress controller runs as normal in the cluster. For more information, see Create a Kubernetes cluster.
- You are connected to a master node by using kubectl. For more information, see Connect to a Kubernetes cluster through kubectl.

- Internet access is required when you pull the image from the address specified in this example. You can replace the address with an image address within your cluster. You can also build and push the image to an image repository and then pull the image from the repository when you use the image.

Step 1: Create a Deployment and a Service

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, select a namespace and click **Create from Template** in the upper-right corner.
6. On the **Create** page, select a sample template or customize a template and click **Create**.

   In this example, two NGINX applications are created: old-nginx and new-nginx.

   The following template is used to create the old-nginx application:

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

The following template is used to create the new-nginx application:

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

7. In the left-side navigation pane of the details page, choose **Network > Services**.

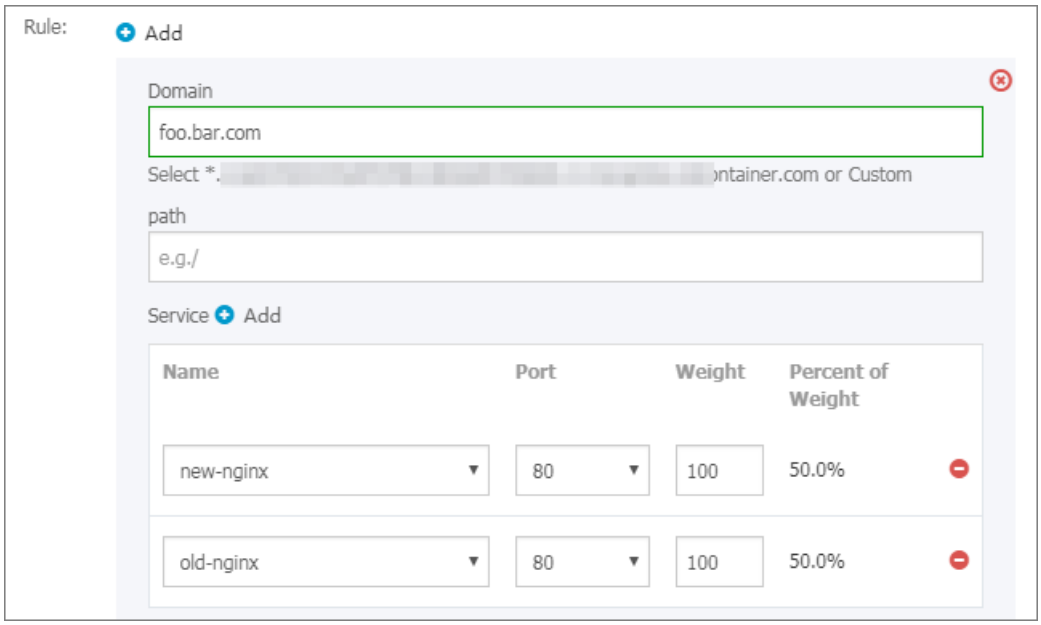
   After the Services are created, you can view the Services on the Services page.

Step 2: Create an Ingress

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. On the details page of the cluster, choose **Network > Ingresses** .
5. On the **Ingresses** page, select a namespace and click **Create** in the upper-right corner.
6. In the dialog box that appears, enter a name for the Ingress. In this example, the Ingress is named nginx-ingress.
7. Configure Ingress rules.

Ingress rules are used to manage external access to Services in the cluster. Ingress rules can be HTTP or HTTPS rules. You can configure the following items in the rules: domain name (virtual hostname), URL path, Service name, port, and weight. For more information, see Ingress configurations.

In this example, a complex rule is added to configure Services for the default domain name and virtual hostname of the cluster. Traffic routing is based on domain names.



Create a simple fanout Ingress that uses multiple domain names

In this example, a virtual hostname is used as the test domain name for external access. Route weights are specified for two backend Services and canary release settings are configured for one of the Services. In a production environment, you can use a domain name that has obtained an Internet Content Provider (ICP) number for external access.

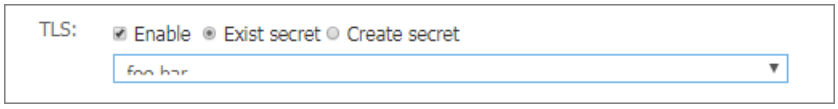
- **Domain**: Enter the test domain name. In this example, the test domain name is `foo.bar.com`.

  You must add the following domain name mapping to the hosts file:

  ```
  118.178.XX.XX foo.bar.com       # The IP address of the Ingress.
  ```

- **Services**: Set the names, paths, port numbers, and weights of the backend Services.
  - Path: Enter the URL of the backend Service. In this example, the root path / is used.
  - Name: In this example, both the old-nginx and new-nginx Services are specified.
  - Port: In this example, port 80 is open.
  - Weight: Set a weight for each backend Service. The weight is a percentage value. The default value is 100. In this example, the weight of each backend Service is set to 50. This means that the two backend Services have the same weight.

8. Configure Transport Layer Security (TLS). Select **EnableTLS** to enable TLS and configure a secure Ingress. For more information, see Configure a secure Ingress.
   - You can use an existing Secret.



a. Log on to a master node. Create a file named *tls.key* and another file named *tls.crt*.

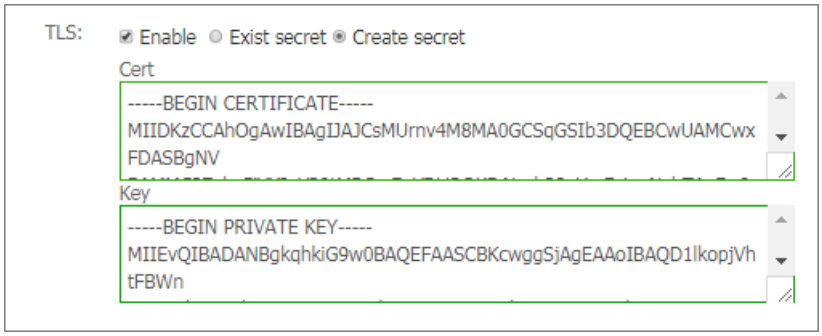
```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt
-subj "/CN=foo.bar.com/O=foo.bar.com"
```

b. Create a Secret.

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

c. Run the `kubectl get secret` command and verify that the Secret is created. Then, you can select the newly created Secret *foo.bar*.

- You can also use the TLS private key and certificate to create a Secret.



a. Log on to a master node, and then create a file named *tls.key* and another file named *tls.crt*.

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

b. Run the `vim tls.key` and `vim tls.crt` commands to obtain the private key and certificate that are generated.

c. Copy the certificate to the Cert field and the private key to the Key field.

9. Configure canary release settings.

> **Note** Only Ingress controllers of 0.12.0-5 and later versions support traffic splitting.

Container Service supports multiple traffic splitting methods. This allows you to select suitable solutions for specific scenarios, such as canary releases and A/B testing:

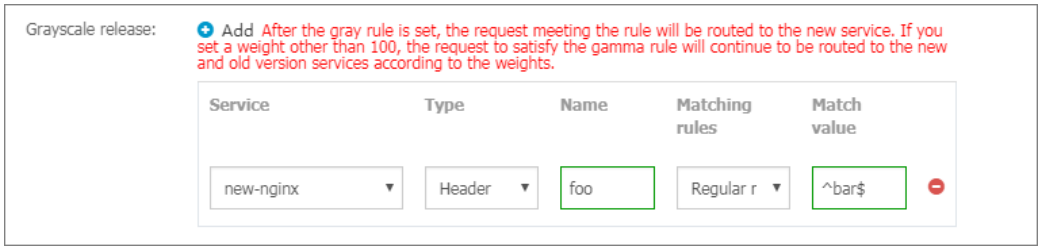
i. Traffic splitting based on request headers

ii. Traffic splitting based on cookies

iii. Traffic splitting based on query parameters

After canary release is configured, only requests that match the specified rules are routed to the new-nginx Service. If the weight of new-nginx is lower than 100%, requests that match the specified rules are routed to the Service based on the Service weight.

In this example, the rule is added to specify that only request headers with headers that match the regular expression `foo=^bar$` are forwarded to new-nginx.



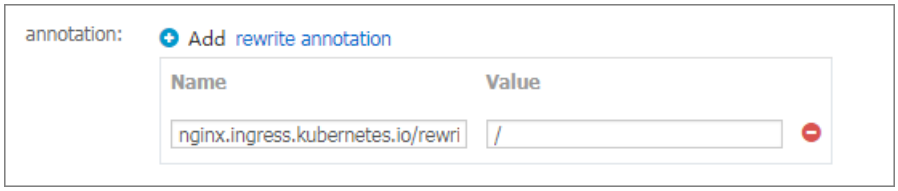
- **Services**: Specify the Services to be accessed.
- **Type**: Select the type of matching rule. Valid values: Header, Cookie, and Query.
- **Name and Match Value**: Specify the names and matching values of custom request fields in key-value pairs.
- **Matching Rule**: Regular expressions and exact matches are supported.

10. Configure annotations.

    Click **Rewrite Annotation** and add an annotation to redirect inbound traffic for the Ingress. For example, `nginx.ingress.kubernetes.io/rewrite-target: /` specifies that */path* is redirected to the root path */* that can be recognized by the backend Services.

    > **Note** In this example, no path is configured for the backend Services. Therefore, you do not need to configure rewrite annotations. Rewrite annotations allow the Ingress to forward traffic through the root path to the backend Services. This avoids the 404 error that is caused by invalid paths.

    You can also click **Add** to enter annotation names and values in key-value pairs. For more information about Ingress annotations, visit https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/.

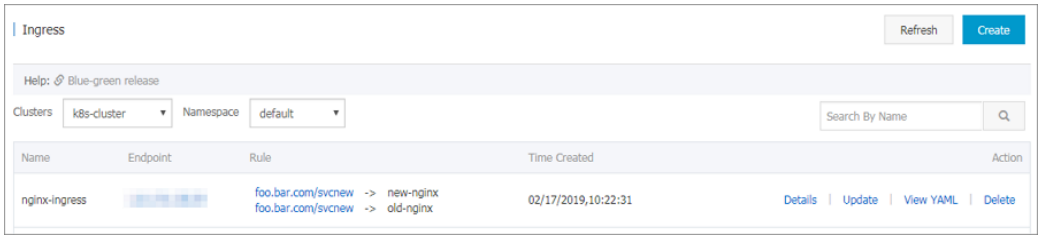


11. Add labels.

    Add labels to describe the characteristics of the Ingress.

12. Click **Create**. You are redirected to the Ingresses page.

    The newly created Ingress appears on the Ingresses page.



13. Click foo.bar.com to visit the NGINX welcome page.

    Click the domain name that points to new-nginx. The old-nginx application page appears.

> **Note** By default, when you enter the domain name in the browser, request headers do not match the regular expression `foo=^bar$`. Therefore, the requests are directed to old-nginx.

14. Log on to a master node by using SSH. Run the following commands to simulate requests with specific headers and check the results:

```
curl -H "Host: foo.bar.com" http://47.107.XX.XX
old
 curl -H "Host: foo.bar.com" http://47.107.XX.XX
old
 curl -H "Host: foo.bar.com" http://47.107.XX.XX                    # Similar to a bro
wser request.
old
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.XX.XX           # Simulate a r
equest with a specific header. The results are returned based on the weight.
new
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.XX.XX
old
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.XX.XX
old
 curl -H "Host: foo.bar.com" -H "foo: bar" http://47.107.XX.XX
new
```

6.8.7. Update an Ingress

You can update Ingresses in the Container Service console.

Prerequisites

- A Kubernetes cluster is created and an Ingress controller is running as normal in the cluster. For more information, see Create a Kubernetes cluster.
- An Ingress is created. For more information, see Create an ingress through the console.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Network > Ingresses**.
5. On the **Ingresses** page, select the namespace, find the Ingress that you want to update, and then click **Update** in the Actions column.
6. In the dialog box that appears, modify the parameters and click **Update**. In this example, `foo.bar`

`.com` is changed to `test.bar.com`.
On the Ingresses page, you can verify that the Ingress rule has changed.

6.8.8. Delete an Ingress

This topic describes how to delete an Ingress.

Prerequisites

- A Kubernetes cluster is created and an Ingress controller is running as normal in the cluster. For more information about how to create a cluster, see Create a Kubernetes cluster.
- An Ingress is created. For more information, see Create an ingress through the console.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Network > Ingresses**.
5. On the **Ingresses** page, select the namespace, find the Ingress that you want to delete, and then click **Delete** in the Actions column.
6. In the message that appears, click **Confirm**.

6.9. Config maps and secrets

6.9.1. Create a ConfigMap

In the Container Service console, you can create a ConfigMap on the ConfigMap page or by using a template. This topic describes how to create a ConfigMap.

Create a ConfigMap on the ConfigMap page

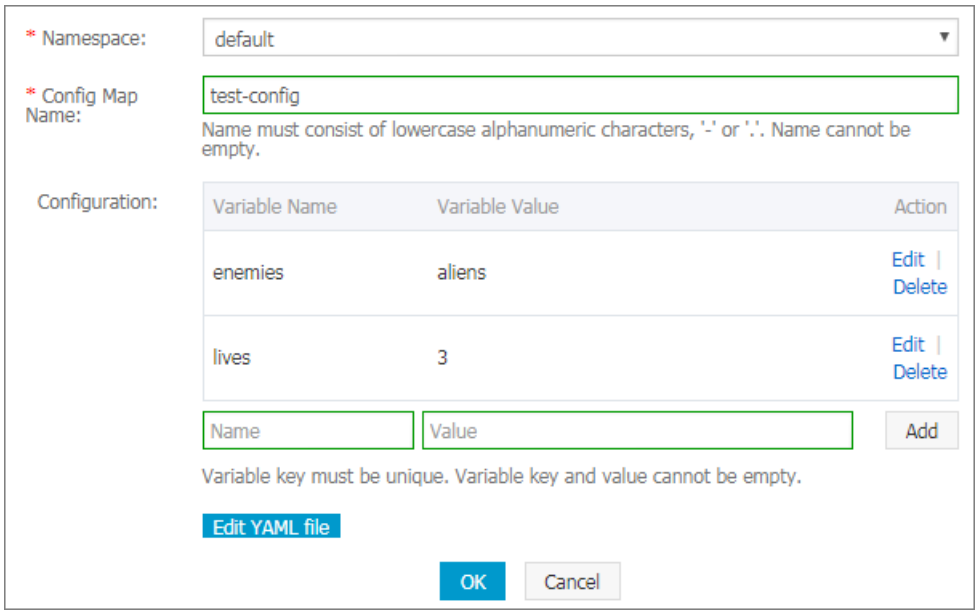
1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
5. On the **ConfigMap** page, select a namespace and click **Create**.
6. Set the parameters and click **OK**.

Create a ConfigMap on the ConfigMap page

Parameter	Description
Clusters	The ID of the cluster that you have selected.

Parameter	Description
Namespaces	The namespace that you have selected. A ConfigMap is a Kubernetes resource object and must be scoped to a namespace.
ConfigMap Name	The name of the ConfigMap. The name can contain lowercase letters, digits, hyphens (-), and periods (.). This parameter is required. Other resource objects must reference the ConfigMap name to obtain the configuration information.
ConfigMap	Specify **Name** and **Value**, and then click **Add** to add the key-value pair. You can also click **Edit YAML file**, modify the parameters in the dialog box that appears, and then click **OK**.

In this example, two variables named enemies and lives are created. Their values are set to aliens and 3 separately.



7. Click **OK**. You can find the test-config ConfigMap on the ConfigMap page.

You can also click **Browse** and upload a configuration file to create a ConfigMap.

Create a ConfigMap from a template

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, select the namespace and click **Create from Template** in the upper-right corner.
6. On the page that appears, set the parameters and click **Create**.

Create a ConfigMap from a template

Parameter	Description
Sample Template	Container Service provides YAML templates of various resource types to help you quickly deploy resource objects. You can select *Custom* from the drop-down list and configure your ConfigMap based on YAML syntax. You can also select the *Resource-ConfigMap* template to create a ConfigMap. In the sample template, the ConfigMap is named aliyun-config and contains two variable files: *game.properties* and *ui.properties*. You can modify the ConfigMap based on your needs.
Template	Enter the template content based on YAML syntax. The template can contain multiple resource objects that are separated by `---`.
Add Deployment	This feature allows you to quickly define a YAML template. You can click **Use Existing Template** to import an existing template.

After the deployment is completed, you can find the ConfigMap named *aliyun-config* on the ConfigMap page.

6.9.2. Use a ConfigMap in a pod

This topic describes how to use a ConfigMap in a pod.

You can use a ConfigMap in a pod in the following scenarios:

- Use a ConfigMap to define environment variables for a pod.
- Use a ConfigMap to set command line parameters.
- Use a ConfigMap in a volume.

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

Limits

To use a ConfigMap in a pod, make sure that the ConfigMap and the pod are in the same cluster and namespace.

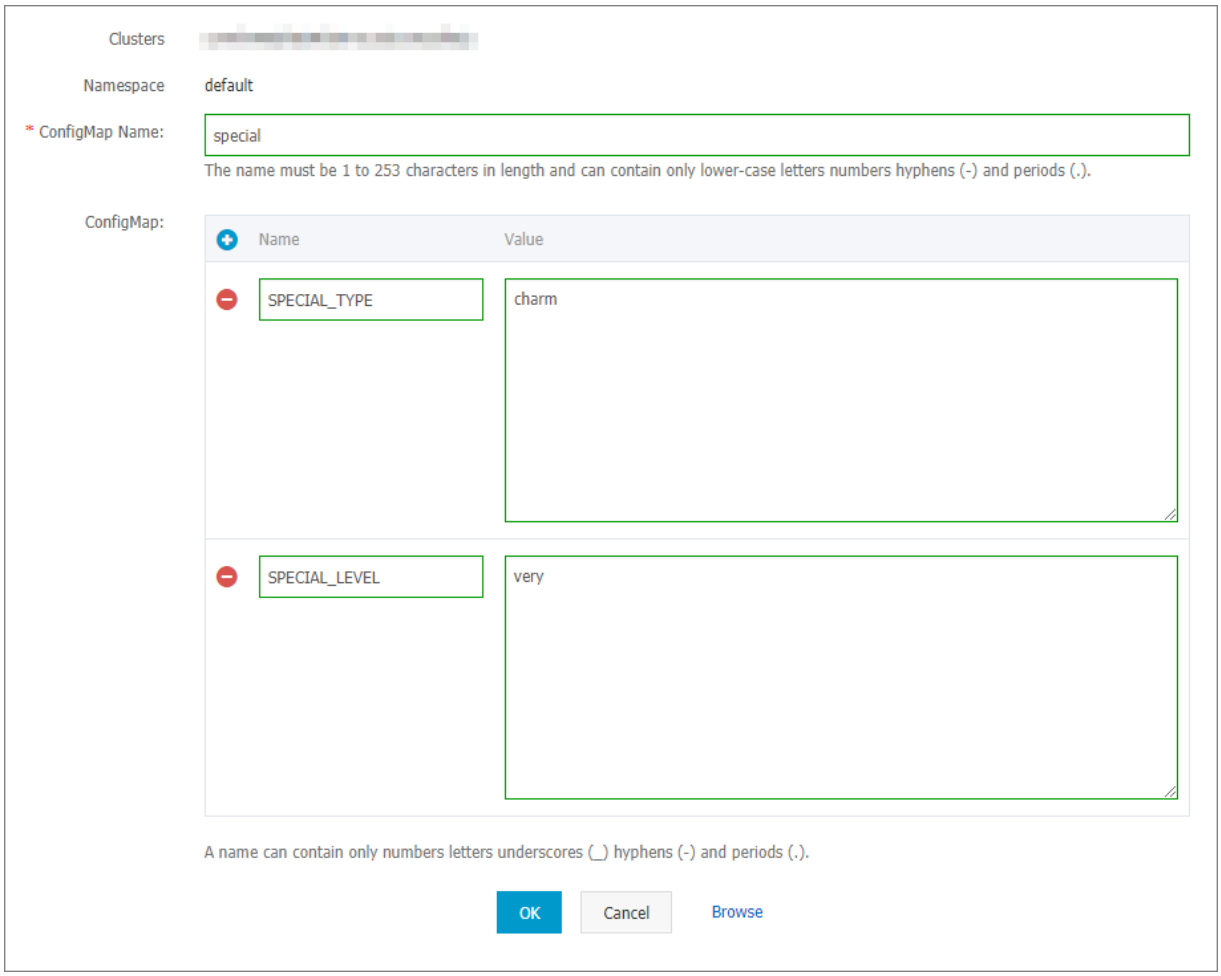
Create a ConfigMap

In this example, a ConfigMap named special_config is created. This ConfigMap consists of two key-value pairs: `SPECIAL_LEVEL: very` and `SPECIAL_TYPE: charm`.

You can use the following YAML template to create the ConfigMap:

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

You can also log on to the Container Service console and choose **Configuration > ConfigMaps** in the left-side navigation pane. You can then click **Create** to create the ConfigMap.



Use a ConfigMap to define one or multiple environment variables for a pod

Use a key-value pair of a ConfigMap to define one environment variable

You can log on to the Container Service console. In the left-side navigation pane, click **Clusters**. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column. In the left-side navigation pane of the details page, choose **Workloads > Deployments**. Click **Create from Template**, select a pod template from the Sample Template drop-down list, and then start the deployment.

You can use the following sample template to create a pod and defines environment variables in the pod. valueFrom is used to reference the value of SPECIAL_LEVEL to define an environment variable.

```
apiVersion: v1
kind: Pod
metadata:
  name: config-pod-1
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:                              ##valueFrom is used to reference the value
of the ConfigMap to define an environment variable.
            configMapKeyRef:
              name: special-config                ##The name of the referenced ConfigMap.
              key: SPECIAL_LEVEL                  ##The key of the referenced key-value pair
.
  restartPolicy: Never
```

To use the values of multiple ConfigMaps to define multiple environment variables, add multiple env parameters to the pod configuration file.

Use all the key-value pairs of a ConfigMap to define multiple environment variables

To define the key-value pairs of a ConfigMap as pod environment variables, you can use the envFrom parameter. The keys in a ConfigMap are used as the names of the environment variables.

The following sample template is used to create a pod:

```
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 of the sepcial-config Con
figMap.
      - configMapRef:
        name: special-config
  restartPolicy: Never
```

Use a ConfigMap to set command line parameters

You can use ConfigMaps to define the commands or parameter values for a container by using the environment variable replacement syntax *$(VAR_NAME)*. The following template is used as an example:

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

After you run the pod, the following output is returned:

```
very charm
```

Use a ConfigMap in a volume

You can use a ConfigMap to define volumes. The following sample template specifies a ConfigMap name under volumes. This stores the key-value pairs of the ConfigMap to the path that you specified in the mountPath field. In this example, the path is /etc/config. This generates configuration files that are named after the keys of the ConfigMap. The corresponding values of the ConfigMap are stored in these files.

```
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 files under the director
y.
      volumeMounts:
        - name: config-volume
          mountPath: /etc/config
   volumes:
     - name: config-volume
       configMap:
         name: special-config
  restartPolicy: Never
```

After you run the pod, the following output is returned:

```
SPECIAL_TYPE
SPECIAL_LEVEL
```

6.9.3. Update a ConfigMap

You can use multiple methods to update a ConfigMap.

Considerations

If you update a ConfigMap, the applications that use the ConfigMap are affected.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
5. On the **ConfigMap** page, select the namespace, find the ConfigMap that you want to update, and then click **Edit** in the Actions column.
6. In the dialog box that appears, modify the configurations and click **OK**.

6.9.4. Delete a ConfigMap

You can use multiple methods to delete a ConfigMap.

Considerations

If you delete a ConfigMap, the applications that use this ConfigMap are affected.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Configurations > ConfigMaps**.
5. On the **ConfigMap** page, select the namespace, find the ConfigMap that you want to delete, and then click **Delete** in the **Actions** column.
6. In the message that appears, click **OK**.

6.9.5. Create a Secret

You can create Secrets for applications in the Container Service console.

Prerequisites

A Kubernetes cluster is created.

Context

We recommend that you use Secrets to store sensitive information in Kubernetes clusters, such as passwords and certificates.

Secrets are classified into the following types:

- Service account: A service account is automatically created by Kubernetes and automatically mounted to the */run/secrets/kubernetes.io/serviceaccount* directory of a pod. The service account provides an identity for the pod to interact with the API server.
- Opaque: This type of secret is encoded in Base64 and used to store sensitive information, such as passwords and certificates.

By default, you can create only Opaque Secrets in the Container Service console. Opaque Secrets store map type data. Therefore, values must be encoded in Base64. You can create Secrets in the Container Service console with a few clicks. Plaintext is automatically encoded in Base64.

You can also create Secrets by using the CLI. For more information, see Kubernetes Secrets.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Configurations > Secrets**.
5. On the **Secrets** page, select the namespace and click **Create** in the upper-right corner.
6. Configure the Secret and click **OK**.

> **Note** To enter Secret data in plaintext, select **Encode Data Values Using Base64**.

Parameter	Description
Name	Enter a name for the Secret. The name must be 1 to 253 characters in length, and can contain only lowercase letters, digits, hyphens (-), and periods (.).
Namespace	Select the namespace of the Secret.
Type	You can select Opaque, Private Repository Logon Secret, or TLS Certificate.
Opaque	If you set Type to Opaque, configure the following parameters: ○ (Optional)To enter Secret data in plaintext, select **Encode Data Values Using Base64**. ○ Configure the Secret in key-value pairs. Click **+ Add**. Enter the keys and values for the Secret in the **Name** and **Value** fields.

Parameter	Description
Private Repository Logon Secret	If you set Type to Private Repository Logon Secret, configure the following parameters: ○ Docker Registry URL: Enter the address of the Docker registry where your Secret is stored. ○ Username: Enter the username that is used to log on to the Docker registry. ○ Password: Enter the password that is used to log on to the Docker registry.
TLS Certificate	If you set Type to TLS Certificate, configure the following parameters: ○ Cert: Enter a TLS certificate. ○ Key: Enter the key for the TLS certificate.

You can view the newly created Secret on the Secrets page.

6.9.6. Modify a Secret

This topic describes how to modify a Secret in the Container Service console.

Prerequisites

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

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Configurations > Secrets**.
5. On the Secrets page, select the namespace, find the Secret that you want to modify, and then click **Edit** in the Actions column.
6. In the dialog box that appears, modify the Secret and click **OK**.

6.9.7. Delete a Secret

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

Prerequisites

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

Context

> **Note** Do not delete Secrets that are generated when the cluster is created.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Configurations > Secrets**.
5. On the **Secrets** page, select the namespace, find the Secret that you want to delete, and then click **Delete** in the **Actions** column.
6. In the message that appears, click **OK**.

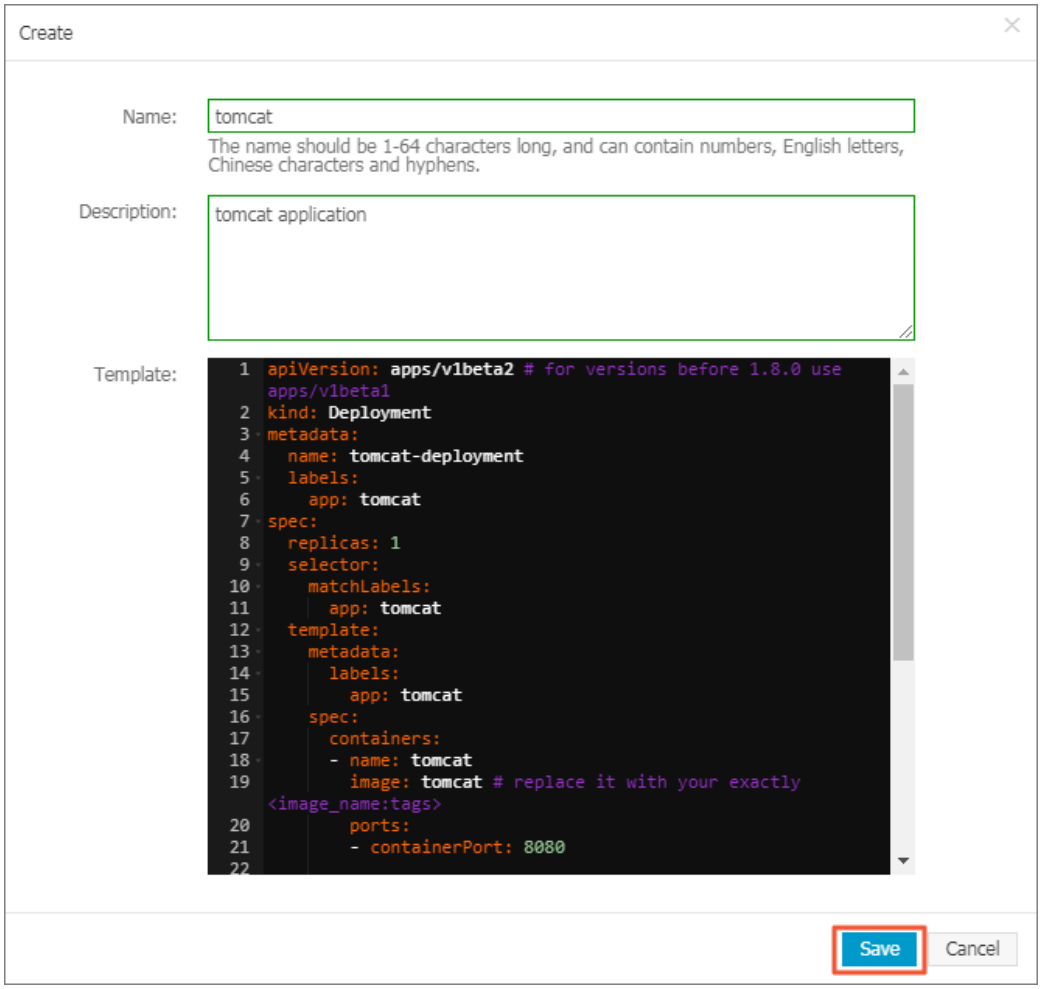
6.10. Templates

6.10.1. Create an orchestration template

This topic describes how to use multiple methods to create orchestration templates through the Container Service console.

Procedure

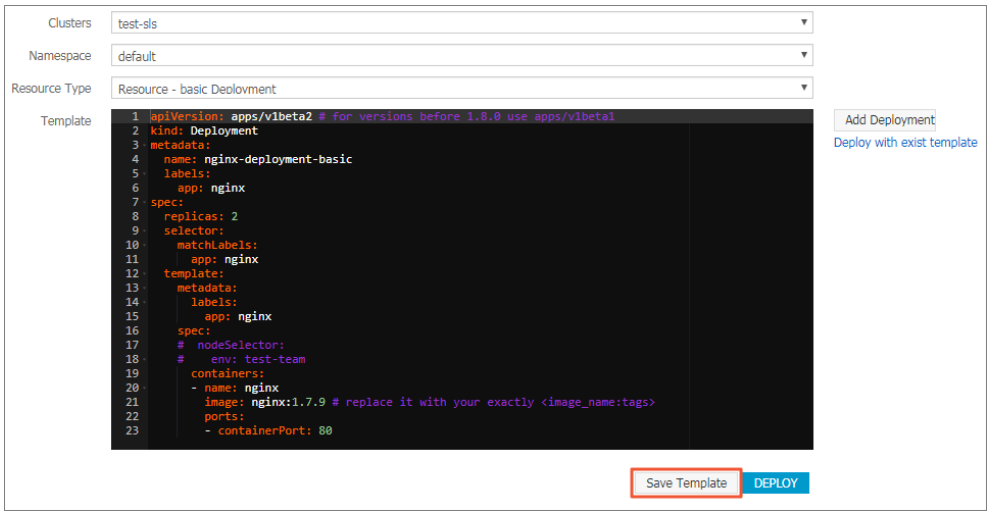
1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > Orchestration Templates** and click **Create** in the upper-right corner.
3. In the dialog box that appears, configure the template, and then click **Save**. This example demonstrates how to create a Tomcat application template that contains a deployment and a service.
   - **Name**: The name of the template.
   - **Description**: Optional. The description of the template.
   - **Template**: Enter the template content based on YAML syntax. The template can contain multiple resource objects that are separated by `---`.



4. After the template is created, you are redirected to the **Templates** page by default. You can find the template on the **My Templates** tab.

5. (Optional)You can also choose **Applications > Deployments** in the left-side navigation pane, and click **Create from Template** to go to the **Create from Template** page. You can modify a built-in template provided by Container Service and save it as a custom template.

i. Select a built-in template and click **Save Template**.



ii. In the dialog box that appears, specify the name, description, and content. Click **Save** to save the template.

> **Note** You can modify the built-in template based on your needs.

iii. In the left-side navigation pane, choose **Marketplace > Orchestration Templates**. You can find the newly created template on the **My Templates** tab.

What's next

You can use the orchestration templates on the **My Templates** tab to quickly create applications.

6.10.2. Update an orchestration template

This topic describes how to edit and update an orchestration template.

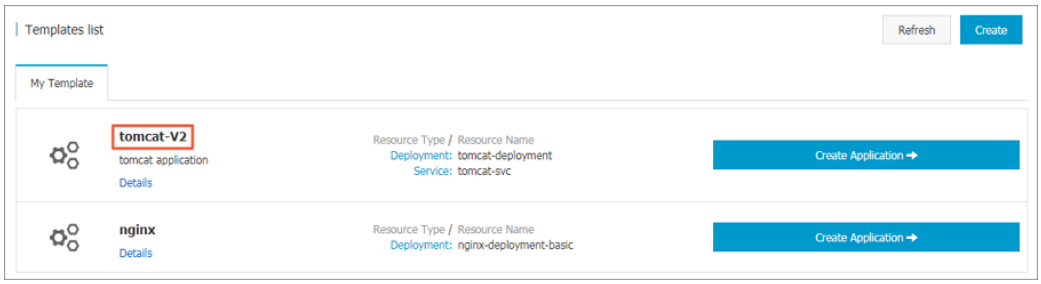
Prerequisites

You have created an orchestration template. For more information, see Create orchestration templates.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > Orchestration Templates**. The **Templates** page appears. You can view existing templates on the **My Templates** tab.
3. Select the target template and click **Details**.
4. On the template details page, click **Edit** in the upper-right corner.
5. In the dialog box that appears, edit the name, description, and template content, and click **Save**.

6. Go to the **Templates** page. You can view the template that you have updated on the **My Templates** tab.



6.10.3. Save an orchestration template as a new one

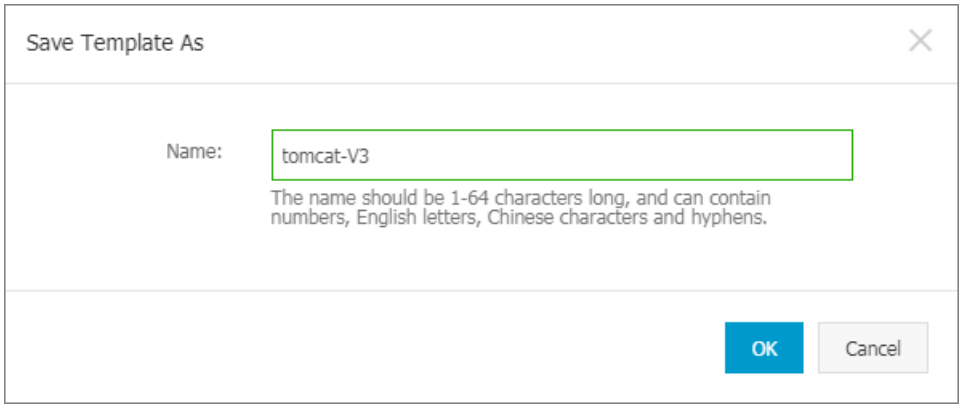
This topic describes how to save an orchestration template as a new one.

Prerequisites

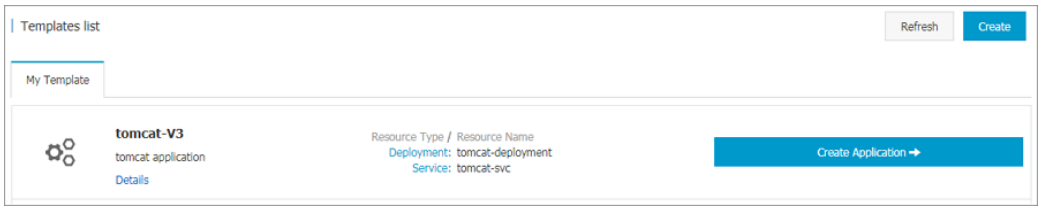
You have created an orchestration template. For more information, see Create orchestration templates.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > Orchestration Templates**. The **Templates** page appears. You can view existing templates on the **My Templates** tab.
3. Select the target template and click **Details**.
4. On the template details page, modify the template and click **Save As** in the upper-right corner.
5. In the dialog box that appears, enter the template name and click **OK**.



6. Go to the **Templates** page. The newly saved template is displayed on the **My Templates** tab.



6.10.4. Download an orchestration template

This topic describes how to download an orchestration template.

Prerequisites

You have created an orchestration template. For more information, see Create orchestration templates.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > Orchestration Templates**. The **Templates** page appears. You can view existing templates on the **My Templates** tab.
3. Select the target template and click **Details**.
4. On the template details page, click **Download** in the upper-right corner to download the template as a YAML file.

6.10.5. Delete an orchestration template

Prerequisites

You have created an orchestration template. For more information, see Create orchestration templates.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > Orchestration Templates**. The **Templates** page appears. You can view existing templates on the **My Templates** tab.
3. Select the target template and click **Details**.
4. On the template details page, click **Delete** in the upper-right corner.
5. In the dialog box that appears, click **OK**.

6.11. Log management

6.11.1. Use Log Service to collect log data from containers

Container Service is integrated with Log Service. When you create a cluster, you can enable Log Service to collect log data from containers, including standard output (stdout) and text files.

Activate Log Service

To activate Log Service, perform the following steps:

1. Log on to the Apsara Uni-manager Management Console. In the top navigation bar, choose **Products > Log Service** to go to the **Log Service** page.
2. Select the required organization and region.
3. Click **SLS** to go to the Log Service console.

Create a Kubernetes cluster that has Log Service enabled

To create a Kubernetes cluster, perform the following steps:

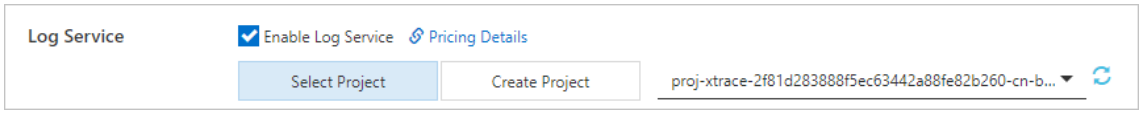
1. Log on to the Container Service console.

   > **Note** The specified organization must be the same as the one that you selected when you activated Log Service. For more information, see Activate Log Service.

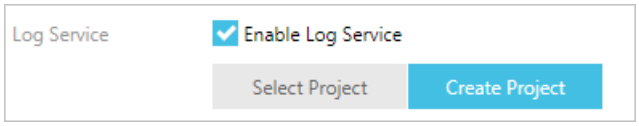
2. In the left-side navigation pane, click **Clusters**.
3. On the Clusters page, click **Create Kubernetes Cluster**. For more information, see Create a Kubernetes cluster.
4. In the lower part of the page, select **Enable Log Service** to install the Log Service component.

   When the **Enable Log Service** check box is selected, the system prompts you to create a Log Service project. Select one of the following methods to specify a project in one of the following ways:

   - You can select an existing project to manage the collected logs.

   - You can click Create Project. Then, a project named `k8s-log-{ClusterID}` is automatically created to manage the collected logs. ClusterID indicates the unique ID of the cluster to be created.

5. Click **Create Cluster** in the upper-right corner of the page.
6. On the **Confirm** page, after all check items are verified, select the terms of service and disclaimer and click **OK** to start the deployment.
   On the Clusters page, you can find the created cluster.

Install the Log Service component in an existing Kubernetes cluster

If you created a Kubernetes cluster and activated Log Service, you can perform the following steps to enable Log Service:

1. Connect to the Kubernetes cluster by using CloudShell.

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

2. Run the *logtail-dedicated.sh* script to install the Log Service component in the Kubernetes cluster.

```
#!/env/bin/bash
yaml=$(cat <<-END
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: alibaba-log-config-file
  namespace: kube-system
data:
```

```
  ilogtail_config.json: |
    {
      "config_server_address" : "http://logtail.$REGION.sls-pub.$INTERNET_DOMAIN",
      "data_server_address" : "http://data.$REGION.sls-pub.$INTERNET_DOMAIN",
      "data_server_list" :
      [
          {
              "cluster" : "$REGION",
              "endpoint" : "data.$REGION.sls-pub.$INTERNET_DOMAIN"
          }
      ],
      "shennong_unix_socket" : false
    }
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: alibaba-log-configuration
  namespace: kube-system
data:
    log-project: "k8s-log-$CLUSTER_ID"
    log-endpoint: "data.$REGION.sls-pub.$INTERNET_DOMAIN"
    log-machine-group: "k8s-group-$CLUSTER_ID"
    log-config-path: "/etc/ilogtail/conf/apsara/ilogtail_config.json"
    log-ali-uid: "$ALI_UID"
    log-access-id: "" # just use blank string
    log-access-key: "" # just use blank string
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: alibaba-log-controller
  namespace: kube-system
  labels:
    k8s-app: alibaba-log-controller
  annotations:
    component.version: "v0.1.3"
    component.revision: "v1"
spec:
  replicas: 1
  template:
    metadata:
      labels:
        k8s-app: alibaba-log-controller
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
    spec:
      serviceAccountName: alibaba-log-controller
      tolerations:
        - operator: "Exists"
      containers:
      - name: alibaba-log-controller
        image: $IMAGE_REPO_URL/acs/log-controller-$ARCH:v0.1.3.0-527ff4d-aliyun
        resources:
```

```
          limits:
            memory: 100Mi
          requests:
            cpu: 50m
            memory: 100Mi
        env:
          - name: "ALICLOUD_LOG_PROJECT"
            valueFrom:
              configMapKeyRef:
                name: alibaba-log-configuration
                key: log-project
          - name: "ALICLOUD_LOG_ENDPOINT"
            valueFrom:
              configMapKeyRef:
                name: alibaba-log-configuration
                key: log-endpoint
          - name: "ALICLOUD_LOG_MACHINE_GROUP"
            valueFrom:
              configMapKeyRef:
                name: alibaba-log-configuration
                key: log-machine-group
          - name: "ALICLOUD_ACS_K8S_FLAG"
            value: "ture"
          - name: "ALICLOUD_ACCESS_KEY_ID"
            valueFrom:
              configMapKeyRef:
                name: alibaba-log-configuration
                key: log-access-id
          - name: "ALICLOUD_ACCESS_KEY_SECRET"
            valueFrom:
              configMapKeyRef:
                name: alibaba-log-configuration
                key: log-access-key
      nodeSelector:
        beta.kubernetes.io/os: linux
---
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: aliyunlogconfigs.log.alibabacloud.com
spec:
  group: log.alibabacloud.com
  version: v1alpha1
  names:
    kind: AliyunLogConfig
    plural: aliyunlogconfigs
  scope: Namespaced
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: alibaba-log-controller
subjects:
- kind: ServiceAccount
  name: alibaba-log-controller
```

```
  name: alibaba-log-controller
  namespace: kube-system
roleRef:
  kind: ClusterRole
  name: alibaba-log-controller
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: alibaba-log-controller
  labels:
    k8s-app: alibaba-log-controller
rules:
- apiGroups: ["log.alibabacloud.com"]
  resources:
  - aliyunlogconfigs
  verbs:
  - update
  - get
  - watch
  - list
- apiGroups: [""]
  resources:
  - configmaps
  verbs:
  - create
  - update
  - get
- apiGroups: [""]
  resources:
  - events
  verbs:
  - create
  - patch
  - update
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: alibaba-log-controller
  namespace: kube-system
  labels:
    k8s-app: alibaba-log-controller
---
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: logtail-ds
  namespace: kube-system
  labels:
    k8s-app: logtail-ds
  annotations:
    component.version: "v0.16.16"
    component.revision: "v0"
```

```
spec:
  updateStrategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        k8s-app: logtail-ds
      annotations:
        scheduler.alpha.kubernetes.io/critical-pod: ''
    spec:
      tolerations:
        - operator: "Exists"
      containers:
      - name: logtail
        image: $IMAGE_REPO_URL/acs/logtail-$ARCH:v0.16.24.0-c46cd2fe-aliyun
        resources:
          limits:
            memory: 512Mi
          requests:
            cpu: 100m
            memory: 256Mi
        livenessProbe:
          exec:
            command:
            - /etc/init.d/ilogtaild
            - status
          initialDelaySeconds: 30
          periodSeconds: 30
        securityContext:
          privileged: false
        env:
          - name: "ALIYUN_LOGTAIL_CONFIG"
            valueFrom:
              configMapKeyRef:
                name: alibaba-log-configuration
                key: log-config-path
          - name: "ALIYUN_LOGTAIL_USER_ID"
            valueFrom:
              configMapKeyRef:
                name: alibaba-log-configuration
                key: log-ali-uid
          - name: "ALIYUN_LOGTAIL_USER_DEFINED_ID"
            valueFrom:
              configMapKeyRef:
                name: alibaba-log-configuration
                key: log-machine-group
          - name: "ALICLOUD_LOG_DOCKER_ENV_CONFIG"
            value: "true"
          - name: "ALICLOUD_LOG_ECS_FLAG"
            value: "ture"
          - name: "ALICLOUD_LOG_DEFAULT_PROJECT"
            valueFrom:
              configMapKeyRef:
                name: alibaba-log-configuration
```

```
            key: log-project
        - name: "ALICLOUD_LOG_ENDPOINT"
          valueFrom:
            configMapKeyRef:
              name: alibaba-log-configuration
              key: log-endpoint
        - name: "ALICLOUD_LOG_DEFAULT_MACHINE_GROUP"
          valueFrom:
            configMapKeyRef:
              name: alibaba-log-configuration
              key: log-machine-group
        - name: "ALICLOUD_LOG_ACCESS_KEY_ID"
          valueFrom:
            configMapKeyRef:
              name: alibaba-log-configuration
              key: log-access-id
        - name: "ALICLOUD_LOG_ACCESS_KEY_SECRET"
          valueFrom:
            configMapKeyRef:
              name: alibaba-log-configuration
              key: log-access-key
        - name: "ALIYUN_LOG_ENV_TAGS"
          value: "_node_name_|_node_ip_"
        - name: "_node_name_"
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        - name: "_node_ip_"
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
      volumeMounts:
      - name: sock
        mountPath: /var/run/docker.sock
      - name: root
        mountPath: /logtail_host
        readOnly: true
      - name: alibaba-log-config-file-volume
        mountPath: /etc/ilogtail/conf/apsara
        readOnly: true
    terminationGracePeriodSeconds: 30
    nodeSelector:
      beta.kubernetes.io/os: linux
    volumes:
    - name: sock
      hostPath:
        path: /var/run/docker.sock
        type: Socket
    - name: root
      hostPath:
        path: /
        type: Directory
    - name: alibaba-log-config-file-volume
      configMap:
```

```
          name: alibaba-log-config-file
END
)
echo "$yaml" > logtail.yml
kubectl create -f logtail.yml
```

3. Replace `<your_server_architecture>`, `<your_k8s_cluster_region_id>`, `<your_k8s_cluster_id>`, `<k8s_cluster_domain_suffix>`, `<your_ali_uid>`, and `<your_image_repo_url>` with actual values, and run the following commands. This allows you to set the environment variables and deploy the component.

```
export ARCH=<your_server_architecture>
export REGION=<your_k8s_cluster_region_id>
export CLUSTER_ID=<your_k8s_cluster_id>
export INTERNET_DOMAIN=<k8s_cluster_domain_suffix>
export IMAGE_REPO_URL=<your_image_repo_url>
export ALI_UID=<your_ali_uid>
bash logtail-dedicated.sh // Run the script to install the component.
```

> **Note**
>
> - `<your_server_architecture>`: the server architecture, for example, amd64.
> - `<your_k8s_cluster_region_id>`: the region where the Kubernetes cluster is deployed, for example, cn-qingdao-apsara-d01.
> - `<your_k8s_cluster_id>`: the ID of the Kubernetes cluster.
> - `<k8s_cluster_domain_suffix>`: the domain suffix of the Kubernetes cluster, for example, env28.internet.com.
> - `<your_ali_uid>`: the ID of the Apsara Stack tenant account, for example, 1234074238634394.
> - `<your_image_repo_url>`: the URL of the image repository, for example, registry.cn-hangzhou.aliyuncs.com.

Create an application and configure Log Service

When you create an application in Container Service, you can configure Log Service to collect logs from containers. You can use only YAML templates to configure Log Service.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**. On the Deployments page, click **Create from Template** in the upper-right corner.
5. Set **Sample Template** to **Custom** and configure the template.

   YAML templates follow the Kubernetes syntax. You can use environment variables to add **collection configurations** and **custom tags**. You must also configure volumeMounts and volumes. The following template is used to create a pod for collecting log data:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  labels:
    app: logtail-test
  name: logtail-test
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: logtail-test
      name: logtail-test
    spec:
      containers:
      - name: logtail
        image: registry.acs.env28.intranet.com/acs/busybox:latest
        args:
        - ping
        - 127.0.0.1
        env:
        - name: aliyun_logs_log-stdout
          value: stdout
        - name: aliyun_logs_log-varlog
          value: /log/*.log
        - name: aliyun_logs_log_tags
          value: tag1=v1
        volumeMounts:
        - name: volumn-sls
          mountPath: /log
      volumes:
      - name: volumn-sls
        emptyDir: {}
```

- Specify the following configurations in order based on your business requirements:
- Use environment variables to add **collection configurations** and **custom tags**. All environment variables for collection configurations must use the `aliyun_logs_` prefix.
- Add log collection configurations in the following format:

  ```
  - name: aliyun_logs_{Logstore name}
    value: {Log path}
  ```

  In the preceding YAML template, two environment variables are added to the log collection configuration. The environment variable `aliyun_logs_log-stdout` specifies that a Logstore named log-stdout is created to store the stdout collected from containers.

  > **Note** The name of the Logstore cannot contain underscores (_). You can use hyphens (-) instead.

- **Custom tags** must be specified in the following format:

```
- name: aliyun_logs_{Tag name without underscores (_)}_tags
  value: {Tag name }={Tag value}
```

After a tag is added, the tag is automatically appended to the log content that are collected from the container.

- If you specify a log path to collect log files that are not stdout, you must configure volumeMounts.

  In the preceding YAML template, the mountPath field in volumeMounts is set to */var/log*. This allows logtail-ds to collect log data from the */var/log/*.log* file.

6. After you modify the YAML template, click **Create** to submit the configurations.

Environment variables

You can specify various environment variables to configure log collection. The following table describes the variables.

Variable	Description	Example	Remarks

aliyun_logs_{key}	• Required.{key} can contain only lowercase letters, digits, and hyphens (-), and cannot contain underscores (_).
	• If the specified aliyun_logs_{key}_logstore does not exist, a Logstore named {key} is created.
	• To collect the stdout of containers, set the value to stdout. You can also set the value to a path inside a container to collect the log files.
		`- name: aliyun_logs_catalina`
		`stdout`
		
		`- name: aliyun_logs_access-log`
		`/var/log/nginx/access.log`
			• By default, the Log Service component collects log files in simple mode. In this case, the collected log data not parsed. If you want to parse the log data, we recommend that you change the collection mode in the Log Service console.
			• The value of {key} must be unique in the cluster.

aliyun_logs_{key}_tags	Optional. This variable is used to add tags to log data. The value must be in the following format: {tag-key}={tag-value}.	`- name: aliyun_logs_catalina_tags` `app=catalina`	-
aliyun_logs_{key}_project	Optional. This variable specifies a project in Log Service. By default, the project that you specified when you created the cluster is used.	`- name: aliyun_logs_catalina_project` `my-k8s-project`	The region of the project must be the same as where logtail-ds is deployed.

Variable	Description	Example	Remarks
aliyun_logs_{key}_logstore	Optional. This variable specifies a Logstore in Log Service. By default, the Logstore is named after {key}.	`- name: aliyun_logs_catalina_tags` `my-logstore`	-
aliyun_logs_{key}_shard	Optional. This variable specifies the number of shards in the Logstore. Valid values: 1 to 10. Default value: 2.	`- name: aliyun_logs_catalina_shard` `4`	-
aliyun_logs_{key}_ttl	Optional. This variable specifies the number of days for which log data is retained. Valid values: 1 to 3650. • To permanently retain log data, set the value to 3650. • Default value: 90.	`- name: aliyun_logs_catalina_ttl` `3650`	-
aliyun_logs_{key}_machinegroup	Optional. This variable specifies the machine group of the application. By default, the machine group is the one where logtail-ds is deployed.	`- name: aliyun_logs_catalina_machinegroup` `my-machine-group`	-

- Scenario 1: Collect logs from multiple applications and store them in the same Logstore

  In this scenario, set the aliyun_logs_{key}_logstore variable. The following example shows how to collect stdout from two applications and store the outputs in stdout-logstore.

  Configure the following environment variables for Application 1:

```
######### Configure environment variables ###########
    - name: aliyun_logs_app1-stdout
      value: stdout
    - name: aliyun_logs_app1-stdout_logstore
      value: stdout-logstore
```

  Configure the following environment variables for Application 2:

```
######### Configure environment variables ###########
    - name: aliyun_logs_app2-stdout
      value: stdout
    - name: aliyun_logs_app2-stdout_logstore
      value: stdout-logstore
```

- Scenario 2: Collect logs from different applications and store them in different projects

  In this scenario, perform the following steps:

  i. Create a machine group in each project and set the machine group ID in the following format: k8s-group-{cluster-id}, where {cluster-id} is the ID of the cluster. You can customize the machine group name.

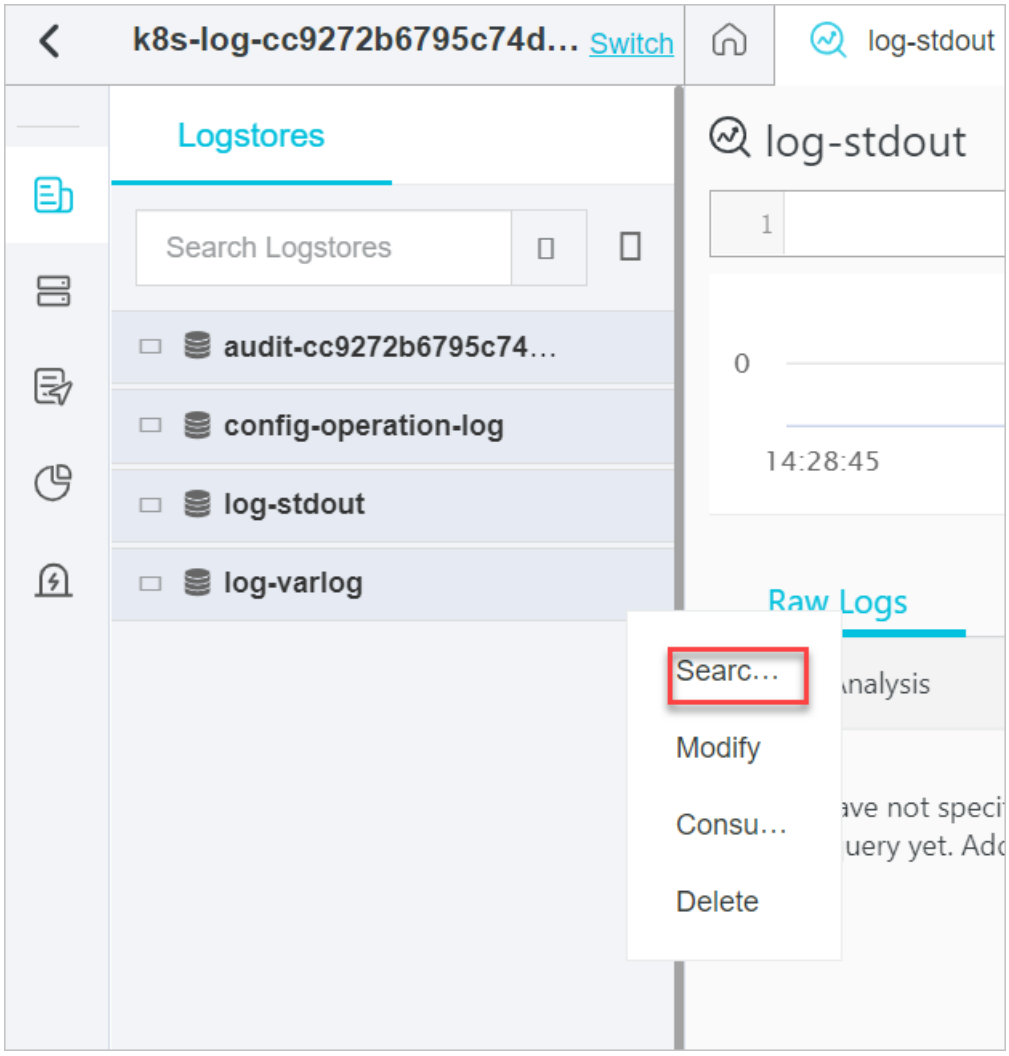
  ii. Specify the project, Logstore, and the created machine group in the environment variables for each application.

```
######### Configure environment variables ###########
    - name: aliyun_logs_app1-stdout
      value: stdout
    - name: aliyun_logs_app1-stdout_project
      value: app1-project
    - name: aliyun_logs_app1-stdout_logstore
      value: app1-logstore
    - name: aliyun_logs_app1-stdout_machinegroup
      value: app1-machine-group
```

View logs

You can view the container log in the Log Service console.

1. Log on to the Log Service console. For more information, see Activate Log Service.
2. Click the project that is associated with the Kubernetes cluster. By default, the project name is in the format of k8s-log-{Kubernetes cluster ID}.
3. In the Logstore list, find the Logstore that is specified when you configure log collection. Move the pointer over the Logstore name and click the  icon. Then, click **Search & Analysis**.
4. you can view the stdout and text log files of the container. You can also find that custom tags are appended to the collected log content.



After Log Service is enabled for the application, you can view the logs of containers in the Container Service console. Perform the following steps to view the logs of containers:

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Operations > Log Center**.
5. On the **Log Center** page, click the **Application Logs** tab and specify the filter conditions. Then, click **Select Logstore** to view the logs of containers.

6.11.2. Configure Log4jAppender for Kubernetes and Log Service

This topic describes how to configure a YAML file to export the log of a Container Service cluster to Log Service, without the need to modify the application. An application that can be managed by calling API operations is deployed in the cluster to generate the log for testing purposes.

Prerequisites

- A Container Service cluster is created. For more information, see Create a Kubernetes cluster.
- An AccessKey pair is created or Resource Access Management (RAM) is activated. Make sure that the required permissions are granted. In this example, an AccessKey pair is created.

Context

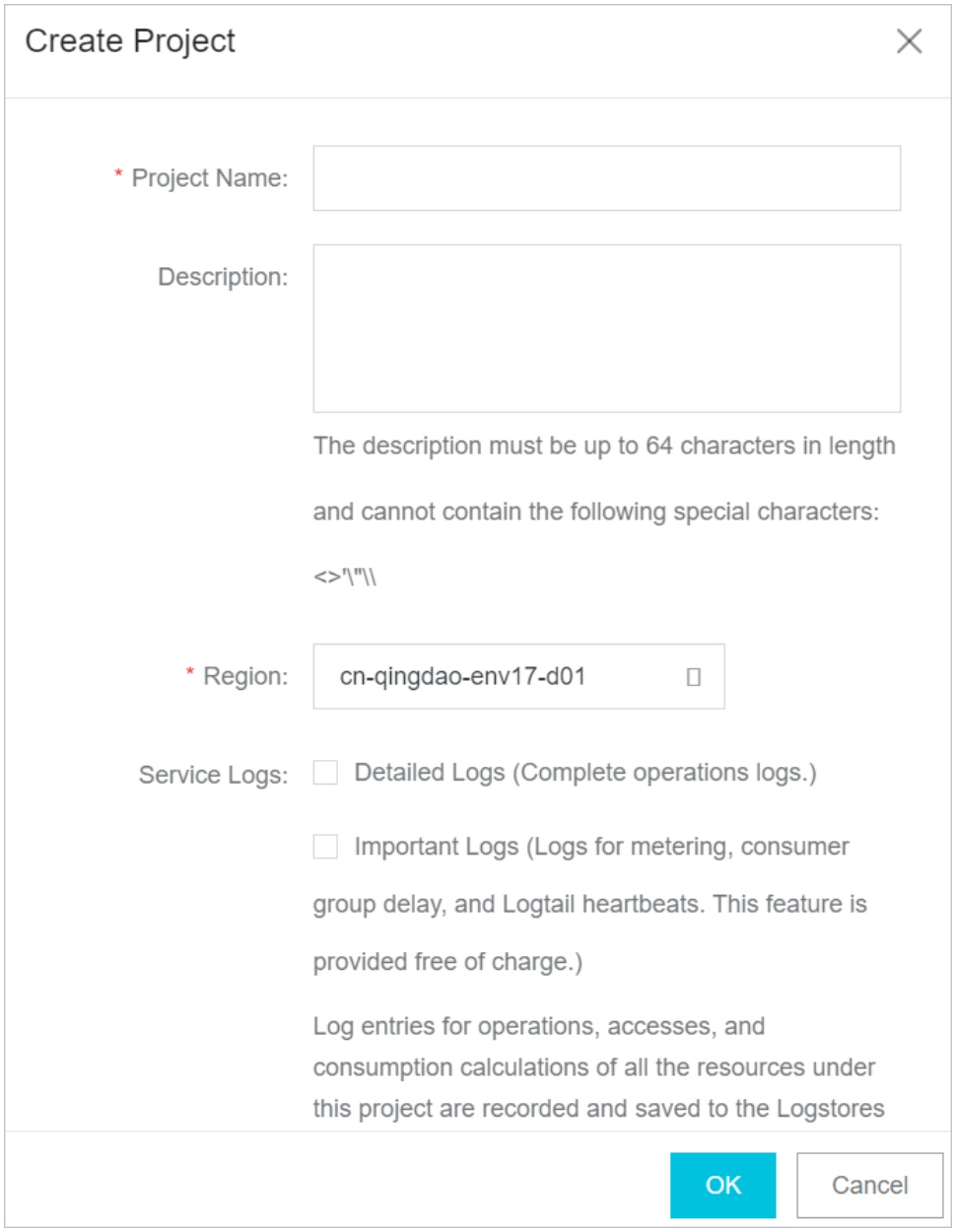
Log4j is an open source project of Apache. Log4j consists of three components: log level, log output destination, and log output format. You can configure Log4jAppender to export log data to the console, a log file, a GUI component, a socket server, an NT event viewer, or a UNIX syslog daemon.

Procedure

1. Configure Log4jAppender in Log Service.
   i. Create a project in Log Service.

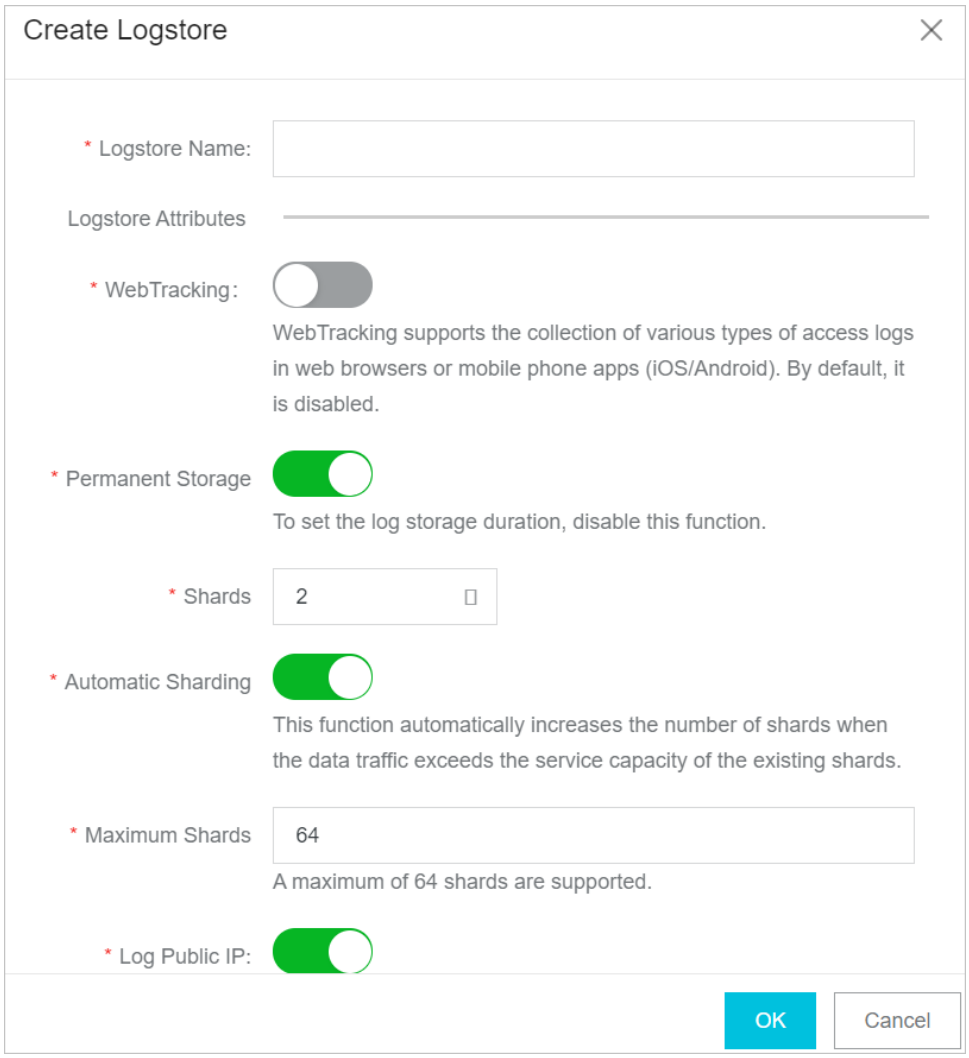
      In this example, a Log Service project named k8s-log4j is created in the same region as your cluster. For more information, see the *Manage projects* chapter of *Log Service User Guide*.

      > **Note** We recommend that you create a Log Service project in the same region as your cluster. When a Log Service project and a cluster are deployed in the same region, the log is transmitted over the internal network. This enables the real-time collection and quick retrieval of log data. This also avoids cross-region transmission, which requires additional bandwidth and time costs.

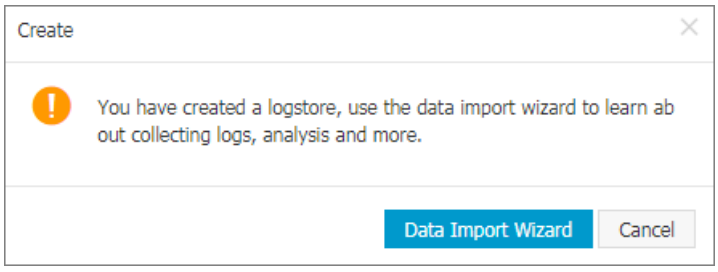

Create Project
* Project Name:
Description:
The description must be up to 64 characters in length and cannot contain the following special characters: <>'\"\\
* Region:
cn-qingdao-env17-d01
Service Logs:
Detailed Logs (Complete operations logs.)
Important Logs (Logs for metering, consumer group delay, and Logtail heartbeats. This feature is provided free of charge.)
Log entries for operations, accesses, and consumption calculations of all the resources under this project are recorded and saved to the Logstores
OK
Cancel

ii. Create a Logstore for the k8s-log4j project.

In this example, a Logstore named k8s-logstore is created. For more information, see the *Manage Logstores* chapter of *Log Service User Guide*.

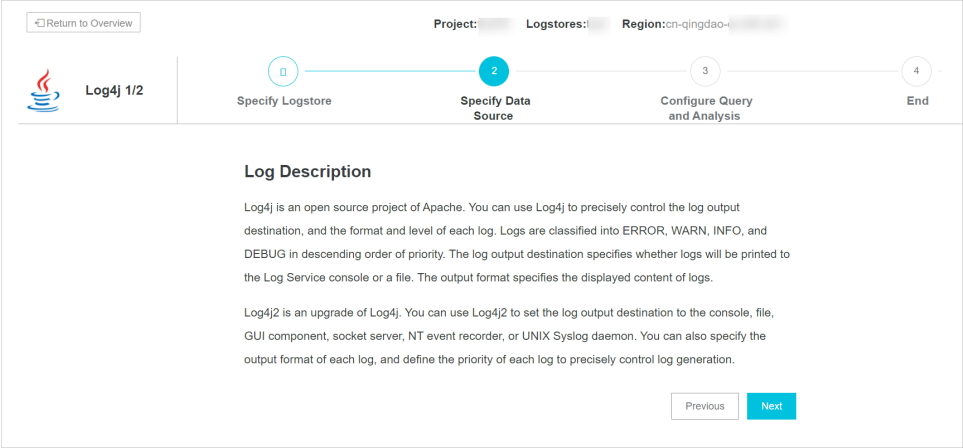


iii. After the k8s-logstore Logstore is created, a message appears, which prompts you to use the Data Import wizard. Click **Data Import Wizard**.



iv. At the top of the **Import Data** dialog box, click the **Custom Code** tab. Then, click Log4j and configure the data import parameters based on the instructions.

In this example, Log4jAppender is configured with the default settings. You can also customize the settings to meet your business requirements.



2. Configure Log4jAppender for the cluster.

In this example, the demo-deployment and demo-Service files are used.

i. Connect to your cluster. For more information, see Connect to a cluster through kubectl.

ii. Obtain the *demo-deployment.yaml* file and configure the `JAVA_OPTS` environment variable.

The following code is a sample template of the *demo-deployment.yaml* file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: log4j-appender-demo-spring-boot
  labels:
    app: log4j-appender
spec:
  replicas: 1
  selector:
    matchLabels:
      app: log4j-appender
  template:
    metadata:
      labels:
        app: log4j-appender
    spec:
      containers:
      - name: log4j-appender-demo-spring-boot
        image: registry.cn-hangzhou.aliyuncs.com/jaegertracing/log4j-appender-demo-
spring-boot:0.0.2
        env:
          - name: JAVA_OPTS                ## Take note of this environment variabl
e.
            value: "-Dproject={your_project} -Dlogstore={your_logstore} -Dendpoint=
{your_endpoint} -Daccess_key_id={your_access_key_id} -Daccess_key={your_access_key_
secret}"
        ports:
        - containerPort: 8080
```

Note The following information is displayed:

- `-Dproject`: the name of your Log Service project. In this example, the name of the project is k8s-log4j.
- `-DlogStore`: the name of your Logstore. In this example, the name of the Logstore is k8s-logstore.
- `-Dendpoint`: the endpoint of Log Service. You must configure the endpoint based on the region of your Log Service project. In this example, the endpoint is cn-hangzhou.log.aliyuncs.com.
- `-Daccess_key_id`: your AccessKey ID.
- `-Daccess_key`: your AccessKey secret.

iii. Run the following command to create a Deployment:

```
kubectl create -f demo-deployment.yaml
```

iv. Obtain the *demo-Service.yaml* file and run the following command to create a Service:

You do not need to modify the configurations in the *demo-Service.yaml* file.

```
kubectl create -f demo-service.yaml
```

3. Generate the log of the Kubernetes cluster log.

You can run the `kubectl get` command to view the deployment status of the related resource objects. After the Deployment and Service are deployed, run the `kubectl get svc` command to check the external IP address of the Service, which is the value of EXTERNAL-IP.

Run the following command:

```
kubectl get svc
```

Output:

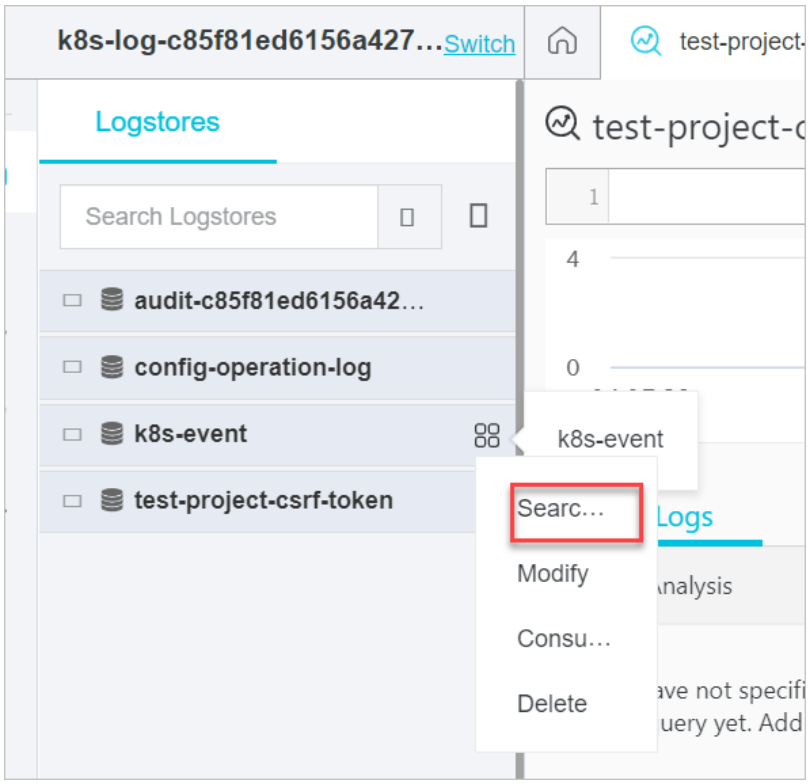
```
NAME                                TYPE           CLUSTER-IP     EXTERNAL-IP      PO
RT(S)          AGE
log4j-appender-demo-spring-boot-svc   LoadBalancer   172.21.XX.XX   120.55.XXX.XXX   80
80:30398/TCP   1h
```

In this example, you can run the `login` command to generate the log of the Kubernetes cluster log. Replace `K8S_SERVICE_IP` in the command with the value of `EXTERNAL-IP`.

> **Note** For a complete list of API operations, see GitHub log4j-appender-demo.

```
curl http://${K8S_SERVICE_IP}:8080/login?name=bruce
```

4. View the log in Log Service
   i. In the **Projects** list, click your Log Service project.
   ii. Click the icon to the right side of the k8s-logstore Logstore and select **Search & Analysis** to view the log of the Kubernetes cluster.



6.12. Monitoring management

6.12.1. Enable ARMS Prometheus

You can view metrics for Container Service clusters on predefined dashboards that are provided by Application Real-Time Monitoring Service (ARMS) Prometheus. This topic describes how to enable ARMS Prometheus in Container Service, how to configure alert rules in ARMS Prometheus, and how to customize monitoring metrics and use Grafana to display monitoring metrics.

Context

ARMS Prometheus is a managed monitoring service that is fully interfaced with the open source Prometheus ecosystem. ARMS Prometheus monitors a wide array of components and provides multiple ready-to-use dashboards. ARMS Prometheus saves you the efforts to manage underlying services, such as data storage, data presentation, and system maintenance.

Enable ARMS Prometheus

Enable ARMS Prometheus when you create a cluster.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**. On the Clusters page, click **Create Kubernetes Cluster** in the upper-right corner.
3. On the **Create Cluster** page, select **Enable Prometheus Monitoring** to the right side of Monitoring Agents. For information about other parameters, see Create a Kubernetes cluster.
4. Click **Create Cluster** in the upper-right corner of the page.
5. On the **Confirm** page, after all check items are verified, select the terms of service and disclaimer and click **OK** to start the deployment.

   After the cluster is created, you can find the cluster on the **Clusters** page in the console.

Enable ARMS Prometheus by installing the ARMS Prometheus component

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, find and click **ack-arms-prometheus**.
4. On the **App Catalog - ack-arms-prometheus** page, select the cluster for which you want to enable ARMS Prometheus in the **Deploy** section, and click **Create**.

View Grafana dashboards provided by ARMS Prometheus

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Operations > Prometheus Monitoring**.
5. On the **Prometheus Monitoring** page, click the name of a Grafana dashboard to view the monitoring data.

> **Notice** If this is the first time you log on to the ARMS console, you must authorize the cluster to access ARMS before you can view the monitoring data. If the authorization page does not appear after you log on to the ARMS Prometheus console, the cluster is authorized to access ARMS.

6.12.2. Monitor application performance

Container Service allows you to use Application Real-Time Monitoring Service (ARMS) to monitor Java and PHP applications that are deployed in clusters. ARMS can automatically discover application topologies, generate 3D topologies, discover and monitor API endpoints, and detect abnormal and slow transactions. ARMS provides an efficient method to diagnose and troubleshoot application issues.

Prerequisites

- A Container Service cluster is created. For more information, see Create a Kubernetes cluster.
- ARMS APM application monitoring has been deployed. For more information, see Enable ARMS Prometheus.

Context

Application Real-Time Monitoring Service (ARMS) is an application performance management (APM) service. After you install the ARMS application monitoring agent in a Container Service cluster, you can use ARMS to monitor Java applications in the cluster without code modifications. ARMS allows you to quickly locate abnormal and slow transactions, reproduce the parameters of API calls, detect memory leaks, and discover system bottlenecks. This significantly improves the efficiency of diagnosing and troubleshooting application issues.

Step 1: Create a user and grant permissions to the user

Create a user and grant the user the permissions to access ARMS. This way, arms-pilot can monitor applications by using the user and the permissions of the user.

1. Create a user.
   i. Log on to the Apsara Uni-manager Management Console. For more information, see Log on to the Container Service console.
   ii. In the upper part of the page, click **Enterprise**.
   iii. In the left-side navigation pane, choose **Users > Users** .
   iv. On the **System Users** tab, click **Create a user**.
   v. In the **Create a user** dialog box, set the **User name**, **Display name**, **Role**, **Organization**, **Logon policy**, **Phone**, and **Email** parameters. Then, click **OK**. In this example, the username is set to acs-arms-pilot.

      Obtain an AccessKey ID and an AccessKey secret in the **AccessKey Information** dialog box.
2. Add the user to a user group.
   i. In the left-side navigation pane of the Enterprise console, choose **Users > User Groups** .
   ii. On the **User Groups** page, select the user group named acs-arms-pilot-usergoup and click **Add User** in the **Actions** column.
   iii. In the **Add User** dialog box, select acs-arms-pilot, click the icon, and then click **OK**.

3. Create a RAM role.
   i. In the left-side navigation pane, choose **Permissions > Role Permissions**.
   ii. On the **Role Authorization** page, click **Create Custom Role**.
   iii. Below the **Create Custom Role** page, click **Advanced Settings** and then click **Create and Configure a RAM role**.
   iv. On the **User Group Management** tab of the **Manage permissions** wizard page, click **Create RAM Role & Attach Policy**.
   v. In the **Basic Configurations** configuration wizard, configure the **Role name** and **Sharing Scope**, and then click **OK**.
   vi. On the right-side of **Configure Custom Policy** configuration wizard, click **Create Policy**, make the relevant configurations and then click **OK**.

   Some of the configuration items are described as follows:

Configuration Items	Illustrate
Policy Name	This document is configured as *AliyunCSARMSAccess*。

Policy Content	Example:
	`{`
	`  "Version": "1",`
	`  "Statement": [`
	`  {`
	`  "Action": "arms:*",`
	`  "Resource": "*",`
	`  "Effect": "Allow"`
	`  }`
	`  ]`
	`}`

   vii. In the **Configure Custom Policy** configuration wizard, search and select *AliyunCSARMSAccess* , and then click **Next**.
   viii. In the **Configure Trust Policy** configuration wizard, configuration on demand **Trust Organization** and **Trust Cloud Service**, and then click **Next**.
   ix. In the **Preview Policy** configuration wizard, after confirming that the information is correct, click **OK**.

Step 2: install arms-pilot

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, click **ack-arms-pilot**.
4. On the **App Catalog - ack-arms-pilot** page, click the Parameters tab and specify the AccessKey ID and AccessKey secret in the template.

```
25  cluster_id: c686cb23020864d08bc9
26
27  # if your cluster is hosted. please fill aliyun ak/sk
28  accessKey: __ACCESSKEY__
29  accessKeySecret: __ACCESSKEY_SECRET__
30
```

5. In the **Deploy** section, select a cluster and click **Create**.

Step 3: Deploy a Java application

1. Connect to a cluster through kubectl.
2. Create a file named *arms-tomcat-demo.yaml* and copy the following code block to the file:

```
apiVersion: v1
kind: Service
metadata:
  name: arms-tomcat-demo
  labels:
    app: arms-tomcat-demo
spec:
  type: LoadBalancer
  ports:
  - port: 80
    name: http
  - port: 8080
    name: tomcat
  selector:
    app: arms-tomcat-demo
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: arms-tomcat-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: arms-tomcat-demo
  template:
    metadata:
      annotations:
        armsPilotAutoEnable: "on"
        armsPilotCreateAppName: "arms-tomcat-demo"
      labels:
        app: arms-tomcat-demo
    spec:
      containers:
        - name: arms-tomcat-demo
          image: registry.acs.intra.en****.shuguang.com/acs/arms-tomcat-demo:v0.1   # R
eplace the image address with the actual one.
          imagePullPolicy: Always
```

Specify the following annotations in the spec.template.metadata.annotations section to enable arms-pilot:

- `armsPilotAutoEnable`: indicate the status of arms-pilot, set the value to `on` to enable

arms-pilot.

- `armsPilotCreateAppName`: indicate the value to the name of the application.

3. Run the following command to deploy the application:

```
kubectl apply -f arms-tomcat-demo.yaml
```

Step 4: View the monitoring data of the application

Go to the **Deployments** page. Verify that the application is deployed and ARMS Console is displayed in the **Actions** column. Click ARMS Console to view monitoring data.

1. Log on to the Container Service console. In the left-side navigation pane, click **Clusters**.
2. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
3. In the left-side navigation pane of the cluster details page, choose **Workloads > Deployments**.
4. On the **Deployments** page, find the application that you created and click **ARMS Console** in the **Actions** column.

   On the Application Overview page, if the value of **Real Time Instance Count** is larger than 0, acs-arms-pilot is deployed and used to monitor the application.

   In the Prometheus Service console, click **Application Details**. Then, click the **JVM monitoring** tab to view the monitoring data.

6.13. GPU

6.13.1. Create a dedicated Kubernetes cluster with GPU-accelerated nodes

This topic describes how to create a dedicated Kubernetes cluster with GPU-accelerated nodes.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**. On the Clusters page, click **Create Kubernetes Cluster** in the upper-right corner.
3. On the **Dedicated Kubernetes** tab of the **Create Cluster** page, set the following parameters.

Parameter	Description
Cluster Name	Enter a name for the cluster. The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-). **Note** The cluster name must be unique among clusters that belong to the same Alibaba Cloud account.

Parameter	Description
Resource Set	Select a resource set. A resource set is a container used to store resources. Each resource must belong to a resource set. After you select a resource set, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource set.
Region	Select the region where you want to deploy the cluster.
VPC	You can select a VPC from the drop-down list. ◦ If the specified VPC has a NAT gateway, Container Service uses this NAT gateway. ◦ If the VPC does not have a NAT gateway, the system automatically creates one. If you do not want the system to create a NAT gateway, clear **Configure SNAT for VPC**. **Note** If you disallow the system to automatically create a NAT gateway and want the VPC to access the Internet, you must manually associate the VPC with a NAT gateway or create SNAT rules for the VPC.
vSwitch	Select vSwitches. You can select up to three vSwitches that are deployed in different **zones**.
Kubernetes Version	Select a Kubernetes version.
Container Runtime	You can select Docker or Sandboxed-Container.

Parameter	Description
Master Configurations	Set the Instance Type and System Disk parameters: ○ Master Node Quantity: You can add up to three master nodes. ○ Instance Type: You can select multiple instance types. For more information, see *Instance families and instance types* in the *ECS* documentation. ○ System Disk: **SSD Disk**, **Ultra Disk**, and **Enhanced SSD** are supported. **Note** You can select **Enable Backup** to back up disk data.
Worker Instance	You can select **Create Instance** or **Add Existing Instance**.
Worker Configurations	If **Worker Instance** is set to **Create Instance**, set the following parameters: ○ Instance Type: Select GPU-accelerated instance types. For more information, see *Instance families and instance types* in the *ECS* documentation. **Note** Select instance types from instance families whose names start with ecs.gn. ○ Selected Types: The selected instance types are displayed. ○ Quantity: Set the number of worker nodes. ○ System Disk: **SSD Disk**, **Ultra Disk**, and **Enhanced SSD** are supported. **Note** You can select **Enable Backup** to back up disk data. ○ Mount Data Disk: **SSD Disk**, **Ultra Disk**, and **Enhanced SSD** are supported. **Note** ■ You can select **Encrypt Disk** to encrypt disks. ■ You can select **Enable Backup** to back up disk data.

Parameter	Description
Operating System	The CentOS and Alibaba Cloud Linux operating systems are supported.
Password	Set a password that is used to log on to the nodes. **Note** The password must be 8 to 30 characters in length, and must contain at least three of the following types of character: uppercase letters, lowercase letters, digits, and special characters.
Confirm Password	Enter the password again.
Network Plug-in	Flannel and Terway are supported. By default, Flannel is selected.
Pod CIDR Block and Service CIDR	These parameters are optional. For more information, see *Network planning* in *VPC User Guide*. **Note** These parameters are available only when you select an **existing VPC**.
Configure SNAT	This parameter is optional. If you clear Configure SNAT for VPC, you must create a NAT gateway or configure SNAT rules for the VPC.
Security Group	You can select **Create Basic Security Group** or **Create Advanced Security Group**.
Access to the Internet	Specify whether to expose the API server with an elastic IP address (EIP). The Kubernetes API server provides multiple HTTP-based RESTful APIs that can be used to create, delete, modify, query, and watch resource objects such as pods and Services. ◦ If you select this check box, an EIP is created and attached to an internal-facing Server Load Balancer (SLB) instance. Port 6443 used by the API server is exposed on the master nodes. You can connect to and manage the cluster by using kubeconfig files over the Internet. ◦ If you clear this check box, no EIP is created. You can connect to and manage the cluster by using kubeconfig files only from within the VPC.

Parameter	Description
SSH Logon	To enable SSH logon, you must first select Expose API Server with EIP. ○ If you enable SSH logon over the Internet, you can access the cluster by using SSH. ○ If you disable SSH logon over the Internet, you cannot access the cluster by using SSH or kubectl. If you want to access an Elastic Compute Service (ECS) instance in the cluster by using SSH, you must manually bind an EIP to the ECS instance and configure security group rules to open SSH port 22.
Ingress	Specify whether to **Install Ingress Controllers**. By default, **Install Ingress Controllers** is selected.
Log Service	If you enable Log Service, you can select an existing project or create a project. If you select **Enable Log Service**, the Log Service plug-in is automatically installed in the cluster. If you select **Create Ingress Dashboard**, Ingress access logs are collected and displayed on dashboards. By default, **Install node-problem-detector and Create Event Center** is selected.
Monitoring Agents	Select or clear **Enable Prometheus Monitoring**. Prometheus Monitoring provides the basic monitoring of the cluster.
Volume Plug-in	By default, **CSI** is selected.
Deletion Protection	If you select this check box, the cluster cannot be deleted in the console or by calling API operations.
Node Protection	This check box is selected by default to prevent nodes from being deleted in the console or by calling API operations.
Label	Add labels to the cluster.

4. Configure the advanced settings.

Parameter	Description
IP Addresses per Node	The number of IP addresses that can be assigned to a node.

Parameter	Description
Custom Image	You can select a custom image. After you select a custom image, all nodes in the cluster are deployed by using this image.
Kube-proxy Mode	iptables and IPVS are supported. ◦ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the Kubernetes cluster. This mode is suitable for Kubernetes clusters that manage a small number of Services. ◦ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.
Node Port Range	Specify the value of **Node Port Range**.
Taints	Add taints to all worker nodes in the cluster.
Cluster Domain	The default domain name of the cluster is cluster.local. You can specify a custom domain name.
Cluster CA	Specify whether to enable the cluster certification authority (CA) certificate.
User Data	Customize the startup behaviors of ECS instances and import data to the ECS instances. The user data can be used to perform the following operations: ◦ Run scripts during instance startup. ◦ Pass user data as common data into an ECS instance for future reference.

5. Click **Create Cluster** in the upper-right corner of the page.
6. On the **Confirm** page, after all check items are verified, select the terms of service and disclaimer and click **OK** to start the deployment.

6.13.2. Upgrade the NVIDIA driver on a GPU node

This topic describes how to upgrade the NVIDIA driver on a GPU node when workloads are deployed on the node and when no workload is deployed on the node.

Upgrade the NVIDIA driver on a GPU node where workloads are deployed

1. Connect to a Kubernetes cluster through kubectl.
2. Run the following command to set the target node to unschedulable.

```
kubectl cordon node-name
```

> **Note**
>
> - Currently, you can only upgrade the NVIDIA driver on worker nodes.
> - *node-name* must be in the format of *your-region-name.node-id*.
>   - *your-region-name* represents the region where your cluster is deployed.
>   - *node-id* represents the ID of the ECS instance where the target node is deployed.
>
> You can run the following command to query *node-name*.
>
> ```
> kubectl get node
> ```

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

3. Run the following command to migrate pods from the target node to other nodes:

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

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

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

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

5. Run the following command to check the current NVIDIA driver version:

```
nvidia-smi
```

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

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

6. Run the following commands to uninstall the existing driver:

   > **Note**
   > - If your driver version is *384.111*, perform the following steps.
   > - If your driver version is not *384.111*, download the corresponding driver from the official NVIDIA website first.

   ```
   cd /tmp
   ```

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

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

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

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

   ```
   reboot
   ```

8. Download the driver that you want to use from the official NVIDIA website. In this example, version *410.79* is used.
9. Run the following command to install the downloaded driver under the directory where it was saved:

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

10. Run the following commands to configure the driver:

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

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

11. Run the following commands to update device-plugin:

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

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

12. Log on to a master node and run the following command to set the target node to schedulable:

```
kubectl uncordon node-name
```

Result

Run the following command on a master node to check the NVIDIA driver version on the target node. The driver version is now *410.79*.

> **Note** Replace *node-name* with the target node name.

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

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

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

Upgrade the NVIDIA driver on a GPU node where no workload is deployed

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

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

2. Run the following command to check the current NVIDIA driver version:

```
nvidia-smi
```

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

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

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

   > **Note**
   > - If your driver version is *384.111*, perform the following steps.
   > - If your driver version is not *384.111*, download the corresponding driver from the official NVIDIA website first.

   ```
   cd /tmp
   ```

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

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

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

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

   ```
   reboot
   ```

5. Download the driver that you want to use from the official NVIDIA website. In this example, version *410.79* is used.
6. Run the following command to install the downloaded driver under the directory where it was saved:

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

7. Run the following commands to configure the driver:

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

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

Result

Run the following command on a master node to check the NVIDIA driver version on the target node. The driver version is now *410.79*.

> **Note** Replace *node-name* with the target node name.

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

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

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

6.13.3. Use cGPU to enable GPU sharing and scheduling

You can use the cGPU solution to schedule multiple applications to one GPU and isolate the GPU memory and computing power that are allocated to each application. This topic describes how to use cGPU to enable GPU sharing and scheduling.

Prerequisites

- A dedicated Kubernetes cluster with GPU-accelerated nodes is created. For more information, see Create a dedicated Kubernetes cluster with GPU-accelerated nodes.
- The Docker version used by the nodes is 19.03.5 or later. Docker versions earlier than 19.03.5 support GPU sharing but do not support GPU isolation.

Context

A key requirement of GPU sharing among multiple pods is to isolate the GPU memory and computing power that are allocated to each pod. When you run multiple containers on one GPU, the GPU resources are allocated to each container as requested. However, if one container occupies excessive GPU resources, the performance of the other containers may be affected. To address this issue, many solutions have been developed in the computing industry. Technologies, such as NVIDIA virtual GPU (vGPU), NVIDIA Multi-Process Service (MPS), rCUDA, and vCUDA, all contribute to fine-grained GPU resource allocation.

The cGPU solution uses the server kernel driver that is developed by Alibaba Cloud to provide more efficient use of the underlying drivers of NVIDIA GPUs. cGPU provides the following features:

- High compatibility: cGPU is compatible with standard open source solutions, such as Kubernetes and NVIDIA Docker.
- Ease of use: cGPU provides excellent user experience. To replace a Compute Unified Device Architecture (CUDA) library of an AI application, you do not need to recompile the application or create a new container image.
- Stability: cGPU provides stable underlying operations on NVIDIA GPUs. API operations on CUDA

libraries and some private API operations on CUDA Deep Neural Network (cuDNN) are difficult to call.

- Resource isolation: cGPU ensures that the allocated GPU memory and computing power do not affect each other.

cGPU provides a cost-effective, reliable, and user-friendly solution that allows you to enable GPU scheduling and memory isolation.

Step 1: Create a node pool

Create a node pool and add the `cgpu=true` label to the node pool.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the cluster details page, choose **Nodes > Node Pools**.
5. On the **Node Pools** page, click **Create Node Pool**.
6. In the **Create Node Pool** dialog box, configure the node pool and click **Confirm Order**.

   For more information, see Create a Kubernetes cluster. The following list describes some of the parameters:

   - Name: the name of the node pool.
   - Quantity: the initial number of nodes in the node pool. If you do not want to add nodes to the node pool, set this parameter to 0.
   - ECS Label: Add labels to the Elastic Compute Service (ECS) instances.
   - Node Label: Click the  icon to add a label. Set the key to cgpu and the value to true.

7. Click **Submit**.

Step 2: Install ack-cgpu

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, search for ack-cgpu and click ack-cgpu after it appears.
4. On the **App Catalog - ack-cgpu** page, select the cluster where you want to install ack-cgpu in the **Deploy** section and click **Create**.

   > **Note** The default number of master nodes in a dedicated Kubernetes cluster is three. If the cluster has five master nodes, set the value of `masterCount` to `5`.

5. Verify that ack-cgpu is installed.
   i. In the left-side navigation pane, click **Clusters**.
   ii. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
   iii. In the left-side navigation pane of the details page, choose **Workloads > Pods**.

iv. At the top of the **Pods** page, set **Namespace** to kube-system. Enter gpushare in the search box and click the search icon. Check the pod status after it appears. If the **state** of the pod is Running or Completed, it indicates that ack-cgpu is installed.

Step 3: Verify GPU sharing and scheduling

1. Create a StatefulSet.
   i. Log on to the Container Service console.
   ii. In the left-side navigation pane, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
   iv. On the cluster details page, choose **Workloads > StatefulSets**. On the right side of the StatefulSets page, click **Create from Template**.

v. Set **Sample Template** to **Custom**, copy the following code to the template, and then click **Create**.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: cgpu-test
  labels:
    app: cgpu-test
spec:
  replicas: 3
  serviceName: "cgpu-test"
  podManagementPolicy: "Parallel"
  selector:
    matchLabels:
      app: cgpu-test
  template:
    metadata:
      labels:
        app: cgpu-test
    spec:
      containers:
      - name: cgpu-test
        image: registry.acs.intra.env115.shuguang.com/acs/gpushare-sample:tensorflo
w-1.5
        command:
          - python
          - cgpu/main.py
        resources:
          limits:
            aliyun.com/gpu-mem: 2 # Apply for 2 GiB of GPU memory.
```

Replace `registry.acs.intra.env115.shuguang.com` in the image address based on your business requirements. You can replace the image repository with that of the image used in Step .

If you want to use ack-cgpu to allocate GPU memory to a pod, you must specify the amount of GPU memory in the `resources` section. In this example, the amount is 2 GiB.

```
// Other settings are omitted.
......
        resources:
          limits:
            aliyun.com/gpu-mem: 2 # Apply for 2 GiB of GPU memory.
```

2. Check the amount of GPU memory allocated to the container.
   i. Log on to the Container Service console.
   ii. In the left-side navigation pane, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
   iv. In the left-side navigation pane of the details page, choose **Workloads > StatefulSets**.
   v. On the **StatefulSets** page, click the newly created StatefulSet.

vi. On the **Pods** tab, find the first pod and choose **Terminal > Container: cgpu-test** in the **Actions** column.

vii. Run the `nvidia-smi -L` command to check the ID of the GPU used by the container.

```
:~# nvidia-smi -L
GPU 0: Tesla P4 (UUID: GPU-175540e0-4470-f5bf-059c-bbcbaa3e1fff)
```

viii. Run the `nvidia-smi` command to check the amount of GPU memory allocated to the container.

```
nvidia-smi
Tue Jun  8 12:21:53 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.181.07   Driver Version: 418.181.07   CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P4            On   | 00000000:00:07.0 Off |                    0 |
| N/A   46C    P0    23W /  75W |   1949MiB /  2174MiB |      2%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

ix. Perform the preceding steps to check the GPU ID and amount of GPU memory of the second pod.

The results show that both pods have the same GPU ID. Each pod is allocated with 2,174 MiB of GPU memory.

> **Note** Pods have the same GPU ID only when they run on the same node. In this example, the cluster has only one GPU-accelerated node. Therefore, the two pods have to run on the same node.

x. Log on to a master node of the cluster. Run the following command to check the allocation details of the GPU resources of the node:

```
kubectl inspect cgpu
```

Expected output:

```
NAME      IPADDRESS       GPU0(Allocated/Total)  GPU Memory(GiB)
a52120c  192.168.52.120  6/7                    6/7
--------------------------------------------------
Allocated/Total GPU Memory In Cluster:
6/7 (85%)
```

The output shows that the cluster has 7 GiB of GPU memory in total and 6 GiB has been used.

The preceding results indicate that the pods use the same GPU and each pod is allocated with 2,174 MiB of GPU memory. This means that the allocated GPU resources are isolated among pods. The total GPU memory of one GPU is 7,611 MiB. You can run the `nvidia-smi` command on a node to obtain the total GPU memory of a GPU. If the allocated GPU resources are not isolated among pods, the amount of GPU memory allocated to a pod is 7,611 MiB. In this example, the actual amount of GPU memory allocated to a pod is 2,174 MiB, which is equal to 2 GiB as requested by the

pod. This means the isolation takes effect.

6.13.4. GPU scheduling for Kubernetes clusters with GPU-accelerated nodes

This topic describes GPU scheduling for Kubernetes clusters with GPU-accelerated nodes.

Prerequisites

Container Service, Resource Orchestration Service (ROS), and Resource Access Management (RAM) are activated.

> **Note** Container Service uses ROS to deploy applications in Kubernetes clusters. To create a Kubernetes cluster, you must first activate ROS.

Context

Starting from version 1.8, Kubernetes adds support for the following hardware acceleration devices by using device plug-ins: NVIDIA GPUs, InfiniBand devices, and field-programmable gate arrays (FPGAs). GPU solutions developed by the community will be phased out in version 1.10, and removed from the master code in version 1.11. Container Service enables you to use a Kubernetes cluster with GPU-accelerated nodes to run compute-intensive tasks such as machine learning and image processing. You can deploy applications and achieve auto scaling without the need to install NVIDIA drivers or Compute Unified Device Architecture (CUDA) in advance.

The system performs the following operations when a Kubernetes cluster is created:

- Creates Elastic Compute Service (ECS) instances, configures a public key to enable Secure Shell (SSH) logon from master nodes to other nodes, and then configures the Kubernetes cluster through CloudInit.
- Creates a security group that allows access to the virtual private cloud (VPC) over Internet Control Message Protocol (ICMP).
- If you do not specify an existing VPC, the system creates a VPC and a vSwitch and creates SNAT entries for the vSwitch.
- Adds route entries to the VPC.
- Creates a NAT gateway and an elastic IP address (EIP).
- Creates a RAM user and grants it permissions to query, create, and delete ECS instances, and permissions to add and delete disks. The RAM user is also granted full permissions on Server Load Balancer (SLB) instances, CloudMonitor, VPC, Log Service, and Apsara File Storage NAS (NAS). The system also creates an AccessKey pair for the RAM user. The system automatically creates SLB instances, disks, and VPC route entries based on your configuration.
- Creates an internal-facing SLB instance and opens port 6443.
- Creates an Internet-facing SLB instance and open ports 6443, 8443, and 22. If you enable SSH logon when you create the cluster, port 22 is open. Otherwise, port 22 is not open.

Limits

- Kubernetes clusters support only VPCs.
- By default, you can create only a limited amount of cloud resources with each account. You cannot

create clusters if the quota on clusters is reached. Make sure that you have sufficient quota before you create a cluster. To request a quota increase, submit a ticket.

- By default, you can create at most five clusters across all regions with each account. Each cluster can contain at most 40 nodes. To increase the quota of clusters or nodes, submit a ticket.

  > **Note** In a Kubernetes cluster, you can create at most 48 route entries for each VPC. This means that a cluster can contain at most 48 nodes. To increase the quota of nodes, submit a ticket to increase the quota of route entries first.

- By default, you can create at most 100 security groups with each account.
- By default, you can create at most 60 pay-as-you-go SLB instances with each account.
- By default, you can create at most 20 EIPs with each account.

- Limits on ECS instances:

  Only CentOS is supported.

Create a GN5 Kubernetes cluster

GN5 Kubernetes clusters support only Kubernetes 1.12.6-aliyun.1. Kubernetes 1.11.5 is not supported.

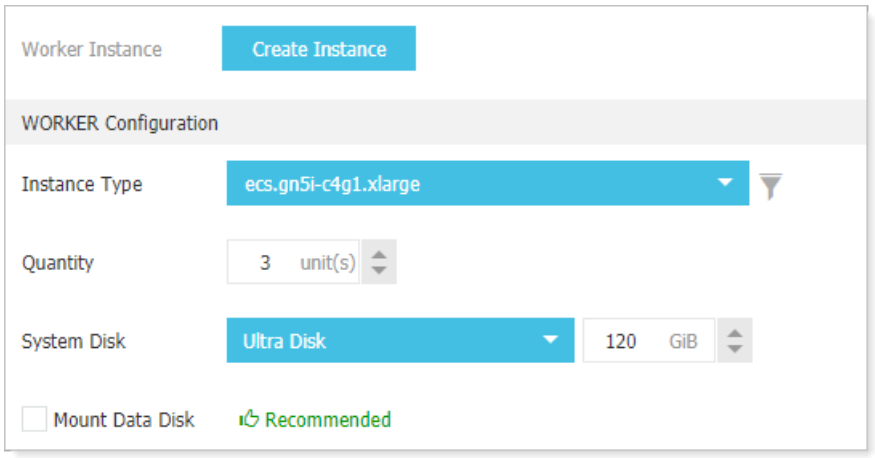
1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. In the upper-right corner of the **Clusters** page, click **Create Kubernetes Cluster**.

   On the cluster configuration page, set the parameters.

   > **Note** To create a cluster with GPU-accelerated nodes, select GPU-accelerated ECS instance types as worker nodes. For more information about other parameters, see Create a Kubernetes cluster.

4. Configure worker nodes. In this example, worker nodes are used to run GPU computing tasks and the gn5i-c4gi1 instance type is selected.

i. If you choose to create worker instances, you must set Instance Type and Quantity. In this example, three GPU-accelerated worker nodes are created.



> **Note** We recommend that you use standard SSDs.

ii. If you choose to add existing instances, you must create GPU-accelerated ECS instances in the region where you want to create the cluster in advance.

5. Specify the other parameters and click **Create Cluster** to start the deployment.

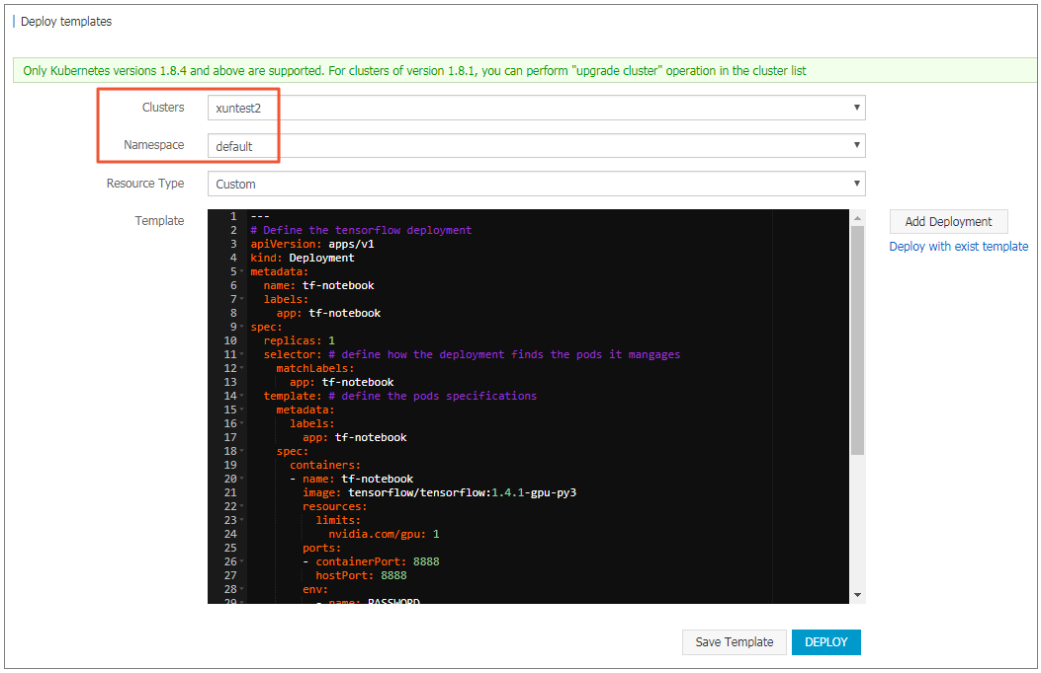
   After the cluster is created, choose **Clusters > Nodes** to go to the Nodes page.

   Select a worker nodes that was configured for the cluster and choose **More > Details** in the Actions column to view the GPU devices that are attached to the node.

Create a GPU experimental environment to run TensorFlow

Jupyter is a standard tool that is used by data scientists to create an experimental environment to run TensorFlow. The following example shows how to deploy a Jupyter application.

1. Log on to the Container Service console. In the left-side navigation pane, choose **Applications > Deployments** to go to the **Deployments** page.
2. In the upper-right corner of the page, click **Create from Template**.
3. Select the required cluster and namespace. Select a sample template, or set Sample Template to Custom and customize the template in the Template field. Then, click **Create** to create the application.

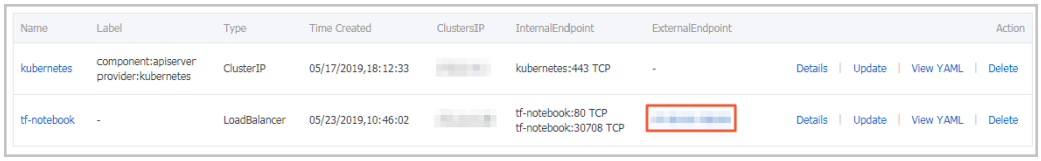



```
---
# Define the tensorflow deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tf-notebook
  labels:
    app: tf-notebook
spec:
  replicas: 1
  selector: # define how the deployment finds the pods it mangages
    matchLabels:
      app: tf-notebook
  template: # define the pods specifications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

In this example, the template uses a Deployment and a Service to create a Jupyter application.

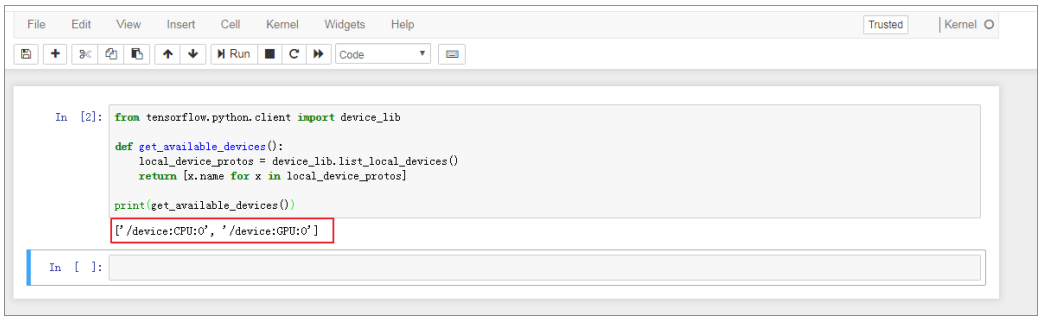
```
---
# Define the tensorflow deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tf-notebook
  labels:
    app: tf-notebook
spec:
  replicas: 1
  selector: # define how the deployment finds the pods it manages
    matchLabels:
      app: tf-notebook
  template: # define the pods specifications
    metadata:
      labels:
        app: tf-notebook
    spec:
      containers:
      - name: tf-notebook
        image: tensorflow/tensorflow:1.4.1-gpu-py3
        resources:
          limits:
            nvidia.com/gpu: 1                      #The number of NVIDIA GPUs that are
requested by the application.
        ports:
        - containerPort: 8888
          hostPort: 8888
        env:
          - name: PASSWORD                         #The password that is used to access
the Jupyter application. You can modify the password as required.
            value: mypassw0rd
# Define the tensorflow service
---
apiVersion: v1
kind: Service
metadata:
  name: tf-notebook
spec:
  ports:
  - port: 80
    targetPort: 8888
    name: jupyter
  selector:
    app: tf-notebook
  type: LoadBalancer                           #An SLB instance is used to route intern
al traffic and perform load balancing.
```

4. In the left-side navigation pane, choose **Ingresses and Load Balancing > Services**. Select the required cluster and namespace, find the tf-notebook Service, and then check its external endpoint.



5. To connect to the Jupyter application, enter `http://EXTERNAL-IP` into the address bar of your browser and enter the password specified in the template.
6. You can run the following program to verify that the Jupyter application is allowed to use GPU resources. The program lists all devices that can be used by TensorFlow:

```
from tensorflow.python.client import device_lib
def get_available_devices():
    local_device_protos = device_lib.list_local_devices()
    return [x.name for x in local_device_protos]
print(get_available_devices())
```



6.13.5. Use labels to schedule pods to GPU-accelerated nodes

To use Kubernetes clusters for GPU computing, you must schedule pods to GPU-accelerated nodes. Container Service allows you to schedule pods to specific GPU-accelerated nodes by adding labels to the GPU-accelerated nodes.

Context

When Kubernetes deploys nodes with NVIDIA GPUs, the attributes of these GPUs are discovered and exposed as node labels. These labels have the following benefits:

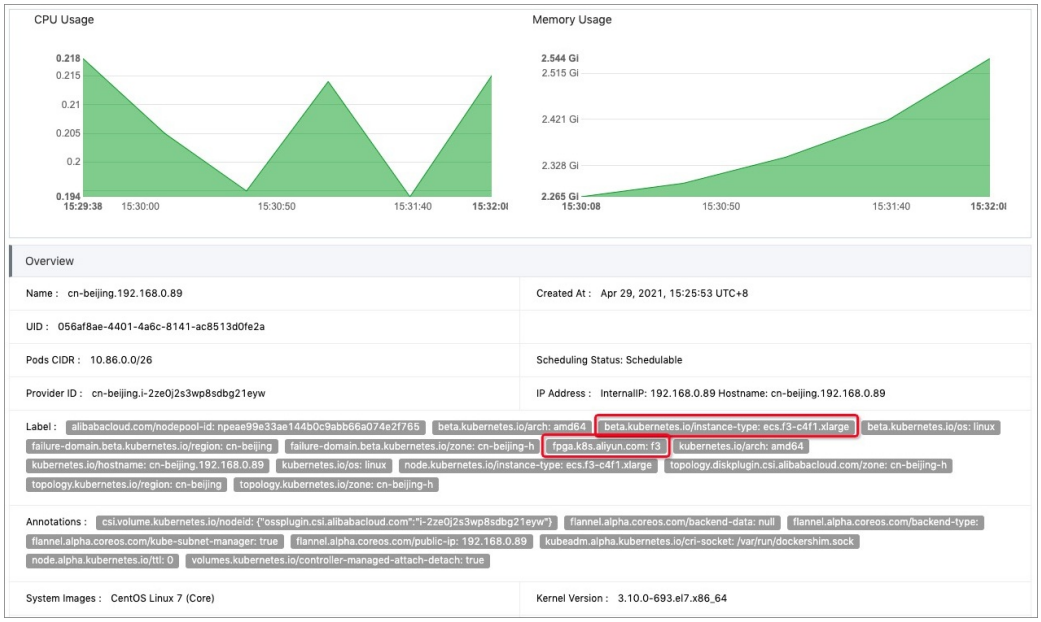
- You can use the labels to filter GPU-accelerated nodes.
- The labels can be used as conditions to schedule pods.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.

> **Note** In this example, the Kubernetes cluster contains three master nodes, among which two are equipped with GPUs. Record the node IP addresses.

5. On the **Nodes** page, select the node that is equipped with GPUs and choose **More > Details** in the Actions column to view the labels that are added to the node.



You can also log on to a master node and run the following command to view the labels of GPU-accelerated nodes:

```
# kubectl get nodes
NAME                                 STATUS    ROLES     AGE       VERSION
cn-beijing.i-2ze2dy2h9w97v65u****    Ready     master    2d        v1.12.6-aliyun.1
cn-beijing.i-2ze8o1a45qdv5q8a****    Ready     <none>    2d        v1.12.6-aliyun.1
# Compare these nodes with the nodes displayed in the console to identify GPU-accelerat
ed nodes.
cn-beijing.i-2ze8o1a45qdv5q8a****    Ready     <none>    2d        v1.12.6-aliyun.1
cn-beijing.i-2ze9xylyn11vop7g****    Ready     master    2d        v1.12.6-aliyun.1
cn-beijing.i-2zed5sw8snjniq6m****    Ready     master    2d        v1.12.6-aliyun.1
cn-beijing.i-2zej9s0zijykp9pw****    Ready     <none>    2d        v1.12.6-aliyun.1
```

Select a GPU-accelerated node and run the following command to query the labels of the node:

```
# kubectl describe node cn-beijing.i-2ze8o1a45qdv5q8a****
Name:               cn-beijing.i-2ze8o1a45qdv5q8a7luz
Roles:              <none>
Labels:             aliyun.accelerator/nvidia_count=1                          #Note
                    aliyun.accelerator/nvidia_mem=12209MiB
                    aliyun.accelerator/nvidia_name=Tesla-M40
                    beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/instance-type=ecs.gn4-c4g1.xlarge
                    beta.kubernetes.io/os=linux
                    failure-domain.beta.kubernetes.io/region=cn-beijing
                    failure-domain.beta.kubernetes.io/zone=cn-beijing-a
                    kubernetes.io/hostname=cn-beijing.i-2ze8o1a45qdv5q8a****
 ......
```

In this example, the following labels are added to the GPU-accelerated node.

key	value
`aliyun.accelerator/nvidia_count`	The number of GPU cores.
`aliyun.accelerator/nvidia_mem`	The size of the GPU memory. Unit: MiB.
`aliyun.accelerator/nvidia_name`	The name of the NVIDIA GPU.

GPU-accelerated nodes of the same type have the same GPU name. You can use this label to locate GPU-accelerated nodes.

```
# kubectl get no -l aliyun.accelerator/nvidia_name=Tesla-M40
NAME                                STATUS    ROLES     AGE       VERSION
cn-beijing.i-2ze8o1a45qdv5q8a****   Ready     <none>    2d        v1.12.6-aliyun.1
cn-beijing.i-2ze8o1a45qdv5q8a****   Ready     <none>    2d        v1.12.6-aliyun.1
```

6. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
7. On the **Deployments** page, select the namespace and click **Create from Template** in the upper-right corner.
8. Create a Deployment for a TensorFlow job. The Deployment is used to schedule pods to a GPU-accelerated node.
9. You can also exclude an application from GPU-accelerated nodes. The following example shows how to schedule a pod based on node affinity for an NGINX application. For more information, see the section that describes node affinity in Create an application from an image.

   The following YAML template is used as an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: not-in-gpu-node
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: aliyun.accelerator/nvidia_name
            operator: DoesNotExist
  containers:
  - name: not-in-gpu-node
    image: nginx
```

Result

In the left-side navigation pane of the details page, choose **Workloads > Pods**. On the Pods page, select the namespace to view pods created in the namespace. On the Pods page, you can find that the pods in the preceding examples are scheduled to the desired nodes. This means that labels can be used to schedule pods to GPU-accelerated nodes.

6.13.6. Manually upgrade the kernel of a GPU node in a cluster

This topic describes how to manually upgrade the kernel of a GPU node in a cluster.

Context

The current kernel version is earlier than `3.10.0-957.21.3` .

Procedure

1. Connect to a Kubernetes cluster through kubectl.
2. Run the following command to set the target GPU node to unschedulable. This example uses node cn-beijing.i-2ze19qyi8votgjz12345 as the target node.

```
kubectl cordon cn-beijing.i-2ze19qyi8votgjz12345
node/cn-beijing.i-2ze19qyi8votgjz12345 already cordoned
```

3. Run the following command to drain the target GPU node:

```
# kubectl drain cn-beijing.i-2ze19qyi8votgjz12345 --grace-period=120 --ignore-daemonset
s=true
node/cn-beijing.i-2ze19qyi8votgjz12345 cordoned
WARNING: Ignoring DaemonSet-managed pods: flexvolume-9scb4, kube-flannel-ds-r2qmh, kube
-proxy-worker-l62sf, logtail-ds-f9vbg
pod/nginx-ingress-controller-78d847fb96-5fkkw evicted
```

4. Uninstall the existing nvidia-driver.

> **Note** This step uninstalls the version 384.111 driver. If your driver version is not 384.111, you need to download a driver from the official NVIDIA website and replace `384.111` with your actual version number.

i. Log on to the target GPU node and run the `nvidia-smi` command to query the driver version.

```
# nvidia-smi -a | grep 'Driver Version'
Driver Version                      : 384.111
```

ii. Run the following commands to download the driver installation package:

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

> **Note** The installation package is required to uninstall the driver.

iii. Run the following commands to uninstall the existing nvidia-driver:

```
chmod u+x NVIDIA-Linux-x86_64-384.111.run
. /NVIDIA-Linux-x86_64-384.111.run --uninstall -a -s -q
```

5. Run the following commands to upgrade kernel:

```
yum clean all && yum makecache
yum update kernel  -y
```

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

```
reboot
```

7. Log on to the GPU node and run the following command to install the kernel-devel package.

```
yum install -y kernel-devel-$(uname -r)
```

8. Run the following commands to download the required driver and install it on the target node. In this example, version 410.79 is used.

```
cd /tmp/
curl -O https://cn.download.nvidia.cn/tesla/410.79/NVIDIA-Linux-x86_64-410.79.run
chmod u+x NVIDIA-Linux-x86_64-410.79.run
sh . /NVIDIA-Linux-x86_64-410.79.run -a -s -q
# warm up GPU
nvidia-smi -pm 1 || true
nvidia-smi -acp 0 || true
nvidia-smi --auto-boost-default=0 || true
nvidia-smi --auto-boost-permission=0 || true
nvidia-modprobe -u -c=0 -m || true
```

9. Check the */etc/rc.d/rc.local* file and check whether the following configurations are included. If not, add the following content.

```
nvidia-smi -pm 1 || true
nvidia-smi -acp 0 || true
nvidia-smi --auto-boost-default=0 || true
nvidia-smi --auto-boost-permission=0 || true
nvidia-modprobe -u -c=0 -m || true
```

10. Run the following commands to restart kubelet and Docker.

```
service kubelet stop
service docker restart
service kubelet start
```

11. Run the following command to set the GPU node to schedulable:

```
# kubectl uncordon cn-beijing.i-2ze19qyi8votgjz12345
node/cn-beijing.i-2ze19qyi8votgjz12345 already uncordoned
```

12. Run the following command on the nvidia-device-plugin container to check the driver version:

```
kubectl exec -n kube-system -t nvidia-device-plugin-cn-beijing.i-2ze19qyi8votgjz12345 n
vidia-smi
Thu Jan 17 00:33:27 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.79       Driver Version: 410.79       CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  On   | 00000000:00:09.0 Off |                    0 |
| N/A   27C    P0    28W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

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

6.14. Auto scaling

6.14.1. Auto scaling of nodes

Container Service provides the auto scaling component to automatically scale the number of nodes. Regular instances and GPU-accelerated instances can be automatically added to or removed from a Container Service cluster to meet your business requirements. This component supports multiple scaling modes, various instance types, and instances that are deployed across zones. This component is applicable to diverse scenarios.

How it works

The auto scaling model of Kubernetes is different from the traditional scaling model that is based on the resource usage threshold. Developers must understand the differences between the two scaling models before they migrate workloads from traditional data centers or other orchestration systems such as Swarm to Kubernetes.

The traditional scaling model is based on resource usage. For example, if a cluster contains three nodes and the CPU utilization or memory usage of the nodes exceeds the scaling threshold, new nodes are added to the cluster. However, you must consider the following issues when you use the traditional scaling model:

- How do you set a proper scaling threshold and how does the system check whether the threshold is exceeded?

  In a Kubernetes cluster, the resource usage of hot nodes is higher than that of other nodes. If you specify the average resource usage as the scaling threshold, scaling activities may not be promptly triggered. If you specify the lowest node resource usage as the scaling threshold, the newly added nodes may not be used. This causes a waste of resources.

- How are the loads balanced after new nodes are added?

  In Kubernetes, pods are the smallest deployable units for applications. Pods are deployed on different nodes in a Kubernetes cluster. When auto scaling is triggered for a cluster or a node in the cluster, pods with high resource usage are not replicated and the resource limits of these pods are not changed. As a result, the loads cannot be balanced to newly added nodes.

- How do you determine whether scaling activities must be triggered and how are scaling activities performed?

  If scale-in activities are triggered based on resource usage, pods that request large amounts of resources but have low resource usage may be evicted. If the number of these pods is large within a Kubernetes cluster, resources may be exhausted and some pods may fail to be scheduled.

How does the auto scaling model of Kubernetes fix these issues? Kubernetes provides a two-layer scaling model that decouples pod scheduling from resource scaling.

In simple terms, pods are scaled based on resource usage. When pods enter the Pending state due to insufficient resources, a scale-out activity is triggered. After new nodes are added to the cluster, the pending pods are automatically scheduled to the newly added nodes. This way, the loads of the application are balanced. The following section describes the auto scaling model of Kubernetes in detail:

- How is a scale-out activity triggered?

  The cluster-autoscaler component scans for pending pods and then triggers scaling activities. When pods enter the Pending state due to insufficient resources, cluster-autoscaler simulates pod scheduling to decide the scaling group that can provide new nodes to accept the pending pods. If a scaling group meets the requirement, nodes from this scaling group are added to the cluster. In simple terms, a scaling group is treated as a node during the simulation. The instance type of the scaling group specifies the CPU, memory, and GPU resources of the node. The labels and taints of the scaling group are also applied to the node. The node is used to simulate the scheduling of the pending pods. If the pending pods can be scheduled to the node, cluster-autoscaler calculates the number of nodes that need to be added from the scaling group.

- How is a scale-in activity triggered?

Only nodes that are added by scaling activities can be removed by cluster-autoscaler. This component cannot manage static nodes. Each node is separately evaluated to determine whether the node needs to be removed. If the resource usage of a node drops below the scale-in threshold, a scale-in activity is triggered for the node. In this case, cluster-autoscaler simulates the eviction of all workloads on the node to determine whether the node can be completely drained. cluster-autoscaler does not drain the nodes that contain specific pods, such as non-DaemonSet pods in the kube-system namespace and pods that are controlled by PodDisruptionBudgets (PDBs). A node is drained before it is removed. After pods on the node are evicted to other nodes, the node can be removed.

- How is a scaling group selected from multiple scaling groups that meet the requirements?

  Different scaling groups are treated as nodes in different specifications. cluster-autoscaler selects a scaling group based on a scoring policy that is similar to a scheduling policy. Nodes are first filtered by the scheduling policy. Among the filtered nodes, the nodes that conform to policies, such as affinity settings, are selected. If no scheduling policy or affinity settings are configured, cluster-autoscaler selects a scaling group based on the least-waste policy. The least-waste policy selects the scaling group that has the fewest idle resources after simulation. If a scaling group of CPU-accelerated nodes and a scaling group of GPU-accelerated nodes both meet the requirements, the scaling group of CPU-accelerated nodes is selected by default.

- How can the success rate of auto scaling be increased? The success rate of auto scaling depends on the following factors:
  - Whether the scheduling policy is met

    After you configure a scaling group, you must be aware of the pod scheduling policies that the scaling group supports. If you are unaware of the pod scheduling policies, you can simulate a scaling activity by using the node selectors of pending pods and the labels of the scaling group.
  - Whether resources are sufficient

    After the scaling simulation is complete, a scaling group is selected. However, the scaling activity fails if the specified types of Elastic Compute Service (ECS) instances in the scaling group are out of stock. To increase the success rate of auto scaling, you can select different types of instances in more than one zone.

- How can auto scaling be accelerated?
  - Method 1: Perform auto scaling in swift mode. After a scaling group experiences a scale-in activity and a scale-out activity, the swift mode is enabled for the scaling group.
  - Method 2: Use custom images that are created from the base image of Alibaba Cloud Linux 2 (formerly known as Aliyun Linux 2). This ensures that the resources of Infrastructure as a Service (IaaS) are delivered 50% faster.

Considerations

- For each account, the default CPU quota for pay-as-you-go instances in each region is 50 vCPUs. You can add at most 48 custom route entries to each route table of a virtual private cloud (VPC). To request a quota increase, submit a ticket.
- The stock of ECS instances may be insufficient for auto scaling if you specify only one ECS instance type for a scaling group. We recommend that you specify multiple ECS instance types with the same specification for a scaling group. This increases the success rate of auto scaling.
- In swift mode, when a node is shut down and reclaimed, the node stops running and enters the *NotReady* state. When a scale-out activity is triggered, the state of the node changes to *Ready*.
- If a node is shut down and reclaimed in swift mode, you are charged only for the disks. This rule does

not apply to nodes that use local disks, such as the instance type of ecs.d1ne.2xlarge, for which you are also charged a computing fee. If the stock of nodes is sufficient, nodes can be launched within a short period of time.

- If elastic IP addresses (EIPs) are associated with pods, we recommend that you do not delete the scaling group or remove ECS instances from the scaling group in the ECS console. Otherwise, these EIPs cannot be automatically released.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the Clusters page, find the cluster that you want to manage and choose **More > Auto Scaling** in the **Actions** column.
4. On the **Configure Auto Scaling** page, set the following parameters and click **Submit**.

Parameter	Description
Cluster	The name of the cluster for which you want to enable auto scaling.
Scale-in Threshold	For a scaling group that is managed by cluster-autoscaler, set the value to the ratio of the requested resources per node to the total resources per node. If the actual value is lower than the threshold, the node is removed from the cluster. **Note** In auto scaling, a scale-out activity is automatically triggered based on node scheduling. Therefore, you need to set only scale-in parameters.
GPU Scale-in Threshold	The scale-in threshold for GPU-accelerated instances. If the actual value is lower than the threshold, the node is removed from the cluster.
Defer Scale-in For	The amount of time that the cluster must wait before the cluster scales in. The default value is 10 minutes.
Cooldown	The cooldown period after a scale-in activity is triggered. No scale-in activity is triggered during the cooldown period. The default value is 10 minutes.

5. Click **Create Scaling Group** and specify the type of resource for auto scaling based on your business requirements. Regular instances and GPU-accelerated instances are supported.
6. In the **Auto Scaling Group Configuration** dialog box, set the following parameters.

Parameter	Description
Region	The region where you want to deploy the scaling group. The scaling group and the Kubernetes cluster must be deployed in the same region. You cannot change the region after the scaling group is created.
VPC	The scaling group and the Kubernetes cluster must be deployed in the same VPC.
VSwitch	The vSwitches of the scaling group. You can specify vSwitches of different zones. The vSwitches allocate pod CIDR blocks to the scaling group.

7. Configure worker nodes.

Parameter	Description
Instance Type	The instance types in the scaling group.
Selected Types	The instance types that you select. You can select at most 10 instance types.
System Disk	The system disk of the scaling group.
Mount Data Disk	Specify whether to mount data disks to the scaling group. By default, no data disk is mounted.
Instances	The number of instances contained in the scaling group. **Note** ◦ Existing instances in the cluster are excluded. ◦ By default, the minimum number of instances is 0. If you specify one or more instances, the system adds the instances to the scaling group. When a scale-out activity is triggered, the instances in the scaling group are added to the cluster to which the scaling group is bound.
Password	Use a password. ◦ Password: Enter the password that is used to log on to the nodes. ◦ Confirm Password: Enter the password again.
Scaling Mode	You can select **Standard** or **Swift**.
RDS Whitelist	The ApsaraDB RDS instances that can be accessed by the nodes in the scaling group after a scaling activity is triggered.

Parameter	Description
Label	Labels are automatically added to nodes that are added to the cluster by scale-out activities.
ECS Label	You can add labels to the selected ECS instances.
Taints	After you add taints to a node, Container Service no longer schedules pods to the node.
CPU Policy	Specify the CPU policy. Valid values: ○ None: indicates that the default CPU affinity is used. This is the default policy. ○ Static: allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.

8. Set advanced options.

Parameter	Description
Custom Security Group	Set a custom security group.
Custom Image	You can select a custom image. Then, all nodes in the Kubernetes cluster are deployed based on the image.
User Data	Customize the startup behaviors of ECS instances and import data to the ECS instances. The user data can be used to perform the following operations: ○ Run scripts during instance startup. ○ Import user data as normal data to an ECS instance for future reference.

9. Click **OK**.

Check the results

In the left-side navigation pane, choose **Applications > Deployments**, select the kube-system namespace. You can find the cluster-autoscaler component. This indicates that the scaling group is created.

FAQ

- Why does the auto scaling component fail to add nodes after a scale-out activity is triggered?

  Check whether the following situations exist:

  ○ The instance types in the scaling group cannot fulfill the resource request from pods. By default, system components are installed for each node. Therefore, the requested pod resources must be less than the resource capacity of the instance type.

- The Resource Access Management (RAM) role does not have the permissions to manage the Kubernetes cluster. You must complete the authorization for each Kubernetes cluster that is involved in the scale-out activity.
- The Kubernetes cluster cannot connect to the Internet. The auto scaling component must call Alibaba Cloud API operations. Therefore, the nodes must have access to the Internet.

- Why does the auto scaling component fail to remove nodes after a scale-in activity is triggered?

  Check whether the following situations exist:

  - The requested resource threshold of each pod is higher than the configured scale-in threshold.
  - Pods that belong to the *kube-system* namespace are running on the node.
  - A scheduling policy forces the pods to run on the current node. Therefore, the pods cannot be scheduled to other nodes.
  - PodDisruptionBudget is set for the pods on the node and the minimum value of PodDisruptionBudget is reached.

  For more information about FAQ, see open source component.

- How does the system choose a scaling group for a scaling activity?

  When pods cannot be scheduled to nodes, the auto scaling component simulates the scheduling of the pods based on the configuration of scaling groups. The configuration includes labels, taints, and instance specifications. If a scaling group can simulate the scheduling of the pods, this scaling group is selected for the scale-out activity. If more than one scaling groups meet the requirements, the system selects the scaling group that has the fewest idle resources after simulation.

6.14.2. Horizontal pod autoscaling

You can create an application that has Horizontal Pod Autoscaling (HPA) enabled in the Container Service console. HPA can automatically scale container resources for your application. You can also use a YAML file to describe HPA settings.

Create an application that has HPA enabled in the Container Service console

Container Service provided by Alibaba Cloud is integrated with HPA. You can create an application that has HPA enabled in the Container Service console. You can enable HPA when you create an application or after the application is created.

Enable HPA when you create an application

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** tab, click **Create from Image**.
6. On the **Basic Information** wizard page, enter a name for your application, set other required parameters, and then click **Next**.

Parameter	Description
Name	Enter a name for the application.
Replicas	The number of pods that are provisioned for the application. Default value: 2.
Type	The type of the application. You can select **Deployments**, **StatefulSets**, **Jobs**, **Cron Jobs**, or **DaemonSets**.
Label	Add a label to the application. The label is used to identify the application.
Annotations	Add an annotation to the application.
Synchronize Timezone	Specify whether to synchronize the time zone between nodes and containers.

7. On the **Container** wizard page, set the container parameters, select an image, and then configure the required computing resources. Click **Next**. For more information, see Configure the containers.

   > **Note** You must configure the required computing resources for the Deployment. Otherwise, you cannot enable HPA.

8. On the **Advanced** wizard page, find the **Access Control** section, click **Create** on the right side of Services, and then set the parameters. For more information, see Create an application from an image.
9. On the **Advanced** wizard page, select **Enable** for **HPA** and configure the scaling threshold and related settings.
   - **Metric**: Select CPU Usage or Memory Usage. The selected resource type must be the same as the one that you have specified in the Required Resources field.
   - **Condition**: Specify the resource usage threshold. HPA triggers scaling activities when the threshold is exceeded.
   - **Max. Replicas**: Specify the maximum number of pods to which the Deployment can be scaled.
   - **Min. Replicas**: Specify the minimum number of pods that must run for the Deployment.
10. In the lower-right corner of the Advanced wizard page, click **Create**. The application is created with HPA enabled.
    Verify the result
    i. Click **View Details** or choose **Workloads > Deployments**. On the page that appears, click the **name of the created application** or click **Details** in the **Actions** column. Then, click the **Horizontal Pod Autoscaler** tab to view information about the scaling group of the application.
    ii. After the application starts to run, container resources are automatically scaled based on the CPU utilization. You can also check whether HPA is enabled in the staging environment by performing a CPU stress test on the pods of the application. Verify that the pods are automatically scaled within 30 seconds.

Enable HPA after an application is created

This example describes how to enable HPA for a stateless application.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage, and click the name of the cluster or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, click the name of the application that you want to manage.
6. Click the **Pod Scaling** tab and click **Create**.
7. In the **Create** dialog box, configure the HPA settings. For more information about how to set the parameters, see HPA settings in Step 9.
8. Click **OK**.

Create an application that has HPA enabled by using kubectl

You can also create a Horizontal Pod Autoscaler by using an orchestration template and associate the Horizontal Pod Autoscaler with the Deployment for which you want to enable HPA. Then, you can run **kubectl** commands to enable HPA.

In the following example, HPA is enabled for an NGINX application.

1. Create a file named *nginx.yml* and copy the following content into the file.

   The following code block is a YAML template that is used to create a Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
        ports:
        - containerPort: 80
        resources:
          requests:                         ##To enable HPA, you must set this paramete
r.
            cpu: 500m
```

2. Run the following command to create an NGINX application:

```
kubectl create -f nginx.yml
```

3. Create a Horizontal Pod Autoscaler.

   Use scaleTargetRef to associate the Horizontal Pod Autoscaler with the Deployment named **nginx**.

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
  namespace: default
spec:
  scaleTargetRef:                              ##Associate the Horizontal Pod Autoscaler
with the nginx Deployment.
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 50
```

   Note You must configure the requested resources for the pods of the application. Otherwise, the Horizontal Pod Autoscaler cannot be started.

4. Run the `kubectl describe hpa name` command. The following output is an example of a warning that is returned:

```
Warning  FailedGetResourceMetric       2m (x6 over 4m)  horizontal-pod-autoscaler  miss
ing request for cpu on container nginx in pod default/nginx-deployment-basic-75675f5897
-mqzs7
Warning  FailedComputeMetricsReplicas  2m (x6 over 4m)  horizontal-pod-autoscaler  fail
ed to get cpu utilization: missing request for cpu on container nginx in pod default/ng
inx-deployment-basic-75675f5
```

5. After the Horizontal Pod Autoscaler is created, run the `kubectl describe hpa name` command.

   If the following output is returned, it indicates that the Horizontal Pod Autoscaler is running as expected:

```
Normal SuccessfulRescale 39s horizontal-pod-autoscaler New size: 1; reason: All metrics
below target
```

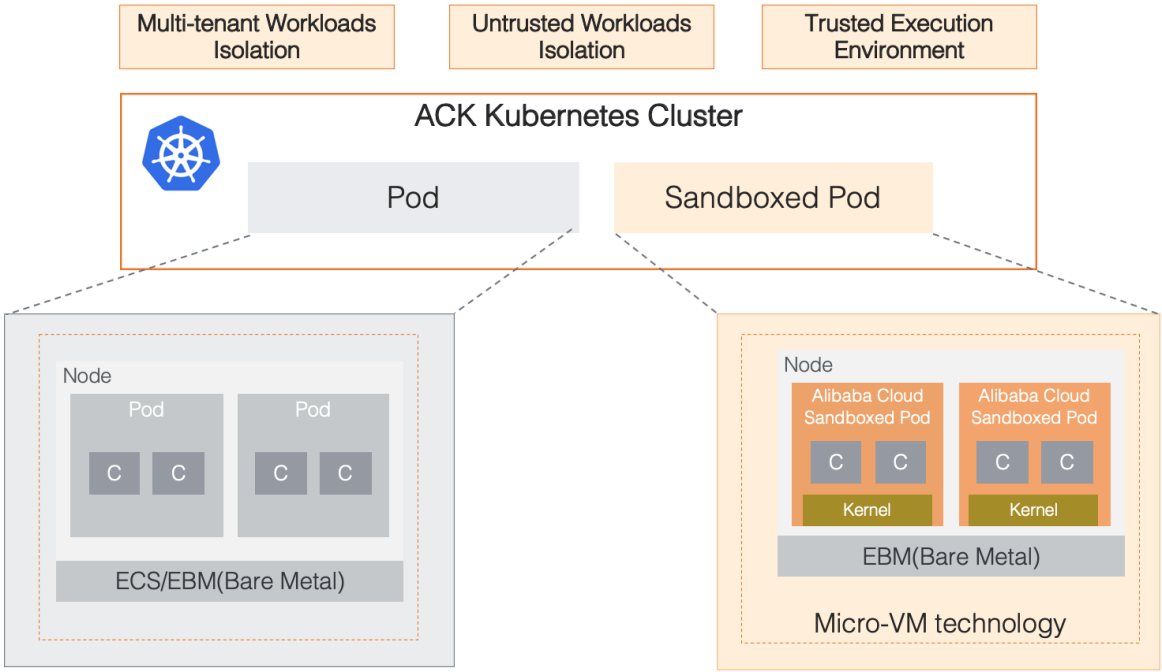
   If the CPU utilization of the NGINX application pod exceeds 50% as specified in the HPA settings, the Horizontal Pod Autoscaler automatically adds pods. If the CPU utilization of the NGINX application pod drops below 50%, the Horizontal Pod Autoscaler automatically removes pods.

6.15. Sandboxed-containers

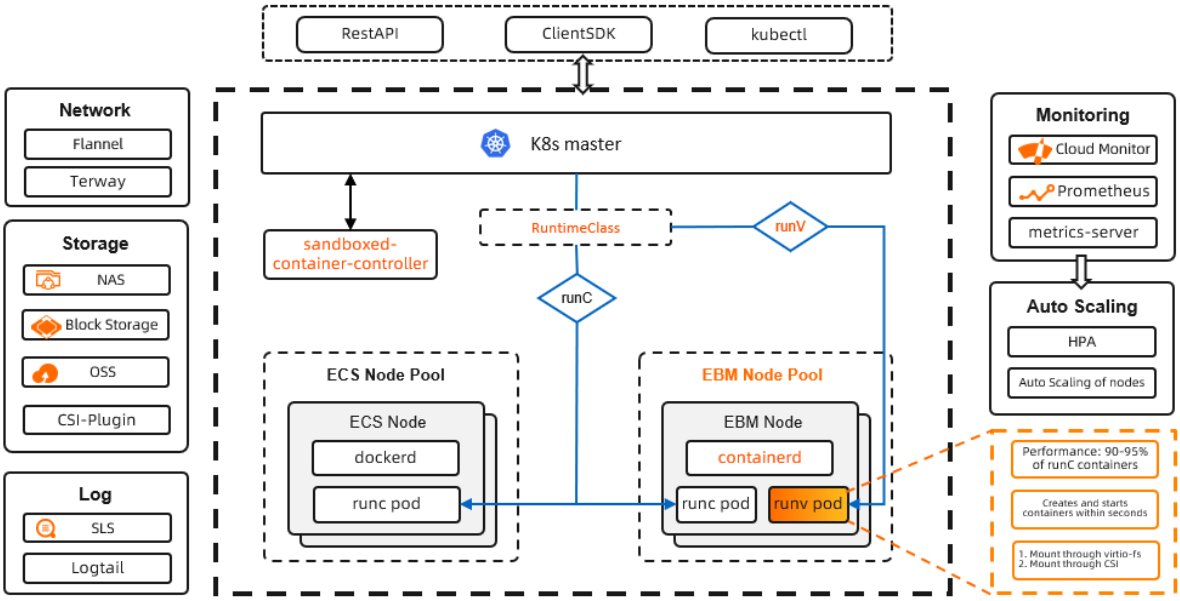
6.15.1. Overview

Sandboxed-Container is an alternative to the Docker runtime. Sandboxed-Container allows you to run applications in a sandboxed and lightweight virtual machine that has a dedicated kernel. This enhances resource isolation and improves security.

Sandboxed-Container is applicable to scenarios such as untrusted application isolation, fault isolation, performance isolation, and workload isolation among multiple users. Sandboxed-Container provides higher security. Sandboxed-Container has minor impacts on application performance and offers the same user experience as Docker in terms of logging, monitoring, and elastic scaling.



Architecture



Features

Sandboxed-Container is a container-securing runtime that is developed by Alibaba Cloud based on sandboxed and lightweight virtual machines. Compared with Sandboxed-Container V1, Sandboxed-Container V2 maintains the same isolation performance and reduces the pod overhead by 90%. It also allows you to start sandboxed containers 3 times faster and increases the maximum number of pods that can be deployed on a host by 10 times. Sandboxed-Container V2 provides the following key features:

- Strong isolation based on sandboxed and lightweight virtual machines.
- Good compatibility with runC in terms of application management.
- Network Attached Storage (NAS) file systems, disks, and Object Storage Service (OSS) buckets can be mounted both directly and through virtio-fs.
- The same user experience as runC in terms of logging, monitoring, and storage.
- Supports RuntimeClass (runC and runV). For more information, see RuntimeClass.
- Easy to use with minimum requirements on technical skills.
- Higher stability than Kata Containers. For more information about Kata Containers, see Kata Containers.

6.15.2. Create a Kubernetes cluster that runs sandboxed containers

This topic describes how to create a Kubernetes cluster that runs sandboxed containers in the Container Service console.

1. Log on to the Container Service console.
2. In the left-side navigation pane of the Container Service console, click **Clusters**. On the Clusters page that appears, click **Create Kubernetes Cluster** in the upper-right corner.
3. On the **Dedicated Kubernetes** tab of the **Create Cluster** page, set the following parameters.

Parameter	Description
Cluster Name	Enter a name for the cluster. The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-). **Note** The cluster name must be unique among clusters that belong to the same Alibaba Cloud account.
Resource Set	Select a resource set. A resource set is a container used to store resources. Each resource must belong to a resource set. After you select a resource set, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource set.
Region	Select the region where you want to deploy the cluster.

Parameter	Description
VPC	You can select a VPC from the drop-down list. ○ If the specified VPC has a NAT gateway, Container Service uses this NAT gateway. ○ If the VPC does not have a NAT gateway, the system automatically creates one. If you do not want the system to create a NAT gateway, clear **Configure SNAT for VPC**. **Note** If you disallow the system to automatically create a NAT gateway and want the VPC to access the Internet, you must manually associate the VPC with a NAT gateway or create SNAT rules for the VPC.
vSwitch	Select vSwitches. You can select up to three vSwitches that are deployed in different zones.
Kubernetes Version	Select a Kubernetes version.
Container Runtime	Select Sandboxed-Container.
Master Configurations	Set the Instance Type and System Disk parameters: ○ Master Node Quantity: You can add up to three master nodes. ○ Instance Type: You can select multiple instance types. For more information, see *Instance families and instance types* in the *ECS* documentation. ○ System Disk: Standard SSDs and ultra disks are supported. **Note** You can select **Enable Backup** to back up disk data.
Worker Instance	You can select **Create Instance** or **Add Existing Instance**.

Parameter	Description
Worker Configurations	If **Worker Instance** is set to **Create Instance**, set the following parameters: ○ Instance Type: Select Elastic Compute Service (ECS) bare metal instance types. ○ Selected Types: The selected instance types are displayed. ○ Quantity: Set the number of worker nodes. ○ System Disk: Standard SSDs and ultra disks are supported. **Note** You can select **Enable Backup** to back up disk data. ○ Mount Data Disk: Standard SSDs and ultra disks are supported. **Note** You can enable disk encryption and data backup when you mount disks. The disks are used to store the root file systems of containers on the nodes. Therefore, you must mount a data disk of at least 200 GiB. We recommend that you mount a data disk of 1 TiB or more.
Password	Set a password that is used to log on to the nodes. **Note** The password must be 8 to 30 characters in length, and must contain at least three of the following types of character: uppercase letters, lowercase letters, digits, and special characters.
Confirm Password	Enter the password again.
Network Plug-in	Flannel and Terway are supported. By default, Flannel is selected.
Pod CIDR Block and Service CIDR	For more information, see *Network planning* in *VPC User Guide*. **Note** These parameters are available only when you select an **existing VPC**.
Configure SNAT	This parameter is optional. If you clear Configure SNAT for VPC, you must create a NAT gateway or configure SNAT rules for the VPC.

Parameter	Description
Access to the Internet	Specify whether to expose the API server with an elastic IP address (EIP). The Kubernetes API server provides multiple HTTP-based RESTful APIs that can be used to create, delete, modify, query, and watch resource objects such as pods and Services. ◦ If you select this check box, an EIP is created and attached to an internal-facing Server Load Balancer (SLB) instance. Port 6443 used by the API server is exposed on the master nodes. You can connect to and manage the cluster by using kubeconfig files over the Internet. ◦ If you clear this check box, no EIP is created. You can connect to and manage the cluster by using kubeconfig files only from within the VPC.
SSH Logon	To enable SSH logon, you must first select Expose API Server with EIP. ◦ If you enable SSH logon over the Internet, you can access the cluster by using SSH. ◦ If you disable SSH logon over the Internet, you cannot access the cluster by using SSH or kubectl. If you want to access an ECS instance in the cluster by using SSH, you must manually bind an EIP to the ECS instance and configure security group rules to open SSH port 22.
Security Group	You can select **Create Basic Security Group** or **Create Advanced Security Group**.
Ingress	Specify whether to install Ingress controllers. By default, **Install Ingress Controller** is selected.
Log Service	If you enable Log Service, you can select an existing project or create a project. If you select **Enable Log Service**, the Log Service plug-in is automatically installed in the cluster. If you select **Create Ingress Dashboard**, Ingress access logs are collected and displayed on dashboards.
Monitoring Agents	Select or clear **Enable Prometheus Monitoring**. Prometheus Monitoring provides the basic monitoring of the cluster.
Volume Plug-in	By default, CSI is selected.
Deletion Protection	If you select this check box, the cluster cannot be deleted in the console or by calling API operations.
Node Protection	This check box is selected by default to prevent nodes from being deleted in the console or by calling API operations.
Label	Add labels to the cluster.

4. Configure the advanced settings.

Parameter	Description
IP Addresses per Node	The number of IP addresses that can be assigned to a node.
Custom Image	You can select a custom image for ECS instances. After you select a custom image, all nodes in the cluster are deployed by using this image.
Kube-proxy Mode	iptables and IPVS are supported. ◦ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the Kubernetes cluster. This mode is suitable for Kubernetes clusters that manage a small number of Services. ◦ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.
Node Port Range	Set the node port range.
Taints	Add taints to all worker nodes in the cluster.
Cluster Domain	The default domain name of the cluster is cluster.local. You can specify a custom domain name.
Cluster CA	Specify whether to enable the cluster certification authority (CA) certificate.
User Data	Customize the startup behaviors of ECS instances and import data to the ECS instances. The user data can be used to perform the following operations: ◦ Run scripts during instance startup. ◦ Pass user data as common data into an ECS instance for future reference.

5. Click **Create Cluster** in the upper-right corner of the page.
6. On the **Confirm** page, click **OK** to start the deployment.

Result

After the cluster is created, you can find the cluster on the **Clusters** page in the console.

6.15.3. Expand a Container Service cluster that runs sandboxed containers

This topic describes how to scale out the worker nodes in a Container Service cluster that runs sandboxed containers in the Container Service console.

Prerequisites

You cannot scale out the master nodes in a Container Service cluster that runs sandboxed containers.

To expand a Container Service cluster that runs sandboxed containers, you must set the parameters as required in the following table. Otherwise, the added nodes cannot run sandboxed containers.

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Expand** in the **Actions** column.
4. Go to the Expand page and set the required parameters.

   In this example, the number of worker nodes in the Container Service cluster is increased from three to five. The following table describes the required parameters.

Parameter	Description
Cluster Name	By default, the name of the Container Service cluster appears.
Region	The region where the Container Service cluster is deployed.
Container Runtime	By default, Sandboxed-Container appears.
VPC	You can select a virtual private cloud (VPC) from the drop-down list. ◦ If the specified VPC is already associated with a NAT gateway, the cluster uses this NAT gateway. ◦ Otherwise, the system automatically creates a NAT gateway. If you do not want the system to create a NAT gateway, clear **Configure SNAT for VPC**. **Note** If you disallow the system to automatically create a NAT gateway and want the VPC to access the Internet, you must manually associate the VPC with a NAT gateway or create Source Network Address Translation (SNAT) rules for the VPC.
VSwitch	Select one or more vSwitches for the cluster. You can select up to three vSwitches that are deployed in different zones.
Billing Method	Only pay-as-you-go nodes are supported.
Existing Worker Nodes	The number of existing workers in the Container Service cluster.
Nodes to Add	Set the number of worker nodes to add.
Worker Nodes After Scaling	The number of worker nodes after the scaling.
Instance Type	Select ECS Bare Metal Instance.
Selected Types	The selected instance types are displayed.

Parameter	Description
System Disk	Standard SSDs and ultra disks are supported. **Note** You can select **Enable Backup** to back up disk data.
Mount Data Disk	Standard SSDs and ultra disks are supported. **Note** You can enable disk encryption and data backup when you mount disks. The disks are used to store the root file systems of containers on the nodes. Therefore, you must mount a disk of at least 200 GiB. We recommend that you mount a disk of at least 1 TiB.
Password	◦ **Password**: Enter the password that is used to log on to the nodes. ◦ **Confirm Password**: Enter the password again.
RDS Whitelist	Set the Apsara RDS whitelist. Add the IP addresses of the nodes in the cluster to the RDS whitelist.
Label	Add labels to the cluster.
Taints	Add taints to all worker nodes in the Kubernetes cluster.
CPU Policy	Specify the CPU policy. Valid values: ◦ None: indicates that the default CPU affinity is used. This is the default policy. ◦ Static: allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
User Data	You can customize the startup behaviors of ECS instances and import data to the ECS instances. The user data can be used to perform the following operations: ◦ Run user data scripts during instance startup. ◦ Import user data as common data to an ECS instance for future reference.

5. Click **Submit**.

What's next

After the Container Service cluster is expanded, go to the details page of the Container Service cluster. In the left-side navigation pane, choose **Clusters > Node Pools**. You can find that the number of worker nodes is increased from 3 to 5.

6.15.4. Create an application that runs in sandboxed containers

This topic describes how to use an image to create an NGINX application that runs in sandboxed containers. The NGINX application is accessible over the Internet.

Prerequisites

A cluster that contains sandboxed containers is created. For more information, see Create a Kubernetes cluster that supports sandboxed containers.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
5. On the **Deployments** page, select the namespace and click **Create from Image** in the upper-right corner.
6. On the **Basic Information** wizard page, specify the basic information of the application and click **Next**.

   Set **Container Runtime** to **runv**. Set the following parameters: **Name**, **Replicas**, **Type**, **Label**, and **Annotations**. Select whether you want to enable **Synchronize Timezone**. The number of replicas specifies the number of pods that are provisioned for in the application.

   > **Note**
   >
   > **Deployments** is selected in this example.

7. Configure containers.

   > **Note** In the upper part of the **Container** wizard page, click **Add Container** to add more containers for the application.

   The following table describes the parameters that are required to configure the containers.

   - General settings

Parameter	Description
Image Name	Click **Select Image**. In the dialog box that appears, select an image and click **OK**. In this example, an NGINX image is selected. You can also enter the address of an image stored in a private registry. The image address must be in the following format: `domainname/namespace/imagename:tag`.

Parameter	Description
Image Version	■ Click **Select Image Version** and select an image version. If you do not specify an image version, the latest image version is used. ■ You can select the following image pulling policies: ■ **ifNotPresent**: If the image that you want to pull is found in the region where the cluster is deployed, Container Service uses the local image. Otherwise, Container Service pulls the image from the corresponding repository. ■ **Always**: Container Service pulls the image from the repository each time the application is deployed or expanded. ■ **Never**: Container Service uses only images on your on-premise machine. **Note** If you select **Image Pull Policy**, no image pulling policy is applied. ■ To pull the image without a Secret, click **Set Image Pull Secret** to set a Secret for pulling images.
Resource Limit	You can specify an upper limit for the CPU, memory, and ephemeral storage space that the container can consume. This prevents the container from occupying an excessive amount of resources. The CPU resource is measured in milicores (one thousandth of one core). The memory resource is measured in MiB. The ephemeral storage resource is measured in GiB.
Required Resources	The amount of CPU and memory resources that are reserved for this application. These resources are exclusive to the container. This prevents the application from becoming unavailable if other services or processes compete for computing resources.
Container Start Parameter	■ stdin: Pass stdin to the container. ■ tty: Stdin is a TeleTYpewriter (TTY).
Privileged Container	■ If you select Privileged Container, privileged=true is set for the container and the privilege mode is enabled. ■ If you do not select Privileged Container, privileged=false is set for the container and the privilege mode is disabled.
Init Container	If you select Init Container, an init container is created. An init container provides tools to manage pods. For more information, see Init Containers.

- (Optional)Ports

  Configure container ports.

  ■ Name: Enter a name for the port.

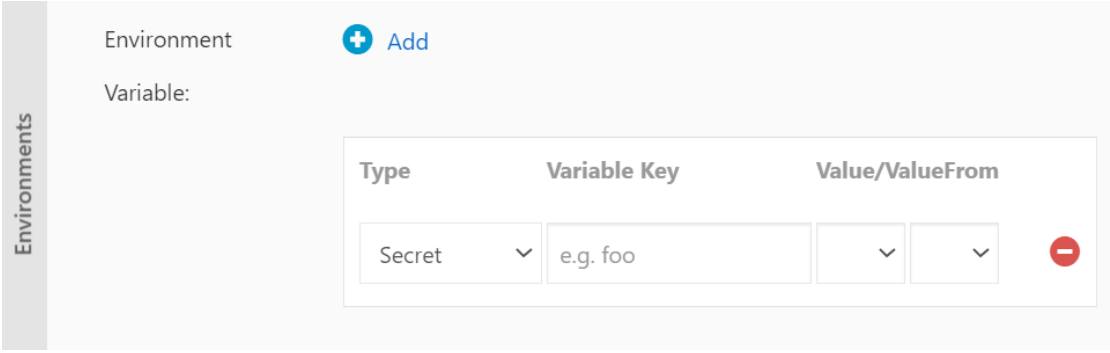
- Container Port: Enter the container port that you want to open. Enter a port number from 1 to 65535.
- Protocol: Select TCP or UDP.

- (Optional)Environments

  You can configure environment variables for pods in key-value pairs. Environment variables are used to apply pod configurations to containers. For more information, see Pod variables.

  - Type: Select the type of the environment variable. You can select **Custom**, **ConfigMaps**, **Secrets**, **Value/ValueFrom**, or **ResourceFieldRef**. If you select ConfigMaps or Secret as the type of the environment variable, all values in the selected ConfigMap or Secret are passed to the container environment variables. In this example, Secret is selected.

    Select **Secrets** from the Type drop-down list and select a Secret from the **Value/ValueFrom** drop-down list. All values in the selected Secret are passed to the environment variable.

    In this case, the YAML file that is used to deploy the application contains the settings that reference all data in the selected Secret.

    ```
    envFrom:
      - secretRef:
          name: test
    ```

  - Variable Key: Specify the name of the environment variable.
  - Value/ValueFrom: Specify the value that is referenced by the environment variable.

- (Optional)Health Check

  Health check settings include liveness and readiness probes. Liveness probes determine when to restart the container. Readiness probes determine whether the container is ready to accept network traffic. For more information about health checks, see Configure Liveness, Readiness, and Startup Probes.

Request type	Description

Request type	Description
HTTP	Sends an HTTP GET request to the container. You can configure the following parameters: ■ Protocol: HTTP or HTTPS. ■ Path: the requested path on the server. ■ Port: Enter the container port that you want to open. Enter a port number from 1 to 65535. ■ HTTP Header: Enter the custom headers in the HTTP request. Duplicate headers are allowed. Key-value pairs are supported. ■ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time (in seconds) that the system must wait before it can send a probe to the container after the container is started. Default value: 3. ■ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ■ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

Request type	Description
TCP	Sends a TCP socket to the container. kubelet attempts to open the socket on the specified port. If the connection can be established, the container is considered healthy. Otherwise, the container is considered unhealthy. You can set the following parameters: ■ Port: Enter the container port that you want to open. Enter a port number from 1 to 65535. ■ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time (in seconds) that the system must wait before it can send a probe to the container after the container is started. Default value: 15. ■ Period (s): the periodSeconds field in the YAML file. This field specifies the time interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ■ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.
Command	Runs a probe command in the container to check the health status of the container. You can set the following parameters: ■ Command: the probe command that is run to check the health status of the container. ■ Initial Delay (s): the initialDelaySeconds field in the YAML file. This field specifies the time (in seconds) that the system must wait before it can send a probe to the container after the container is started. Default value: 5. ■ Period (s): the periodSeconds field in the YAML file. This field specifies the interval (in seconds) at which probes are performed. Default value: 10. Minimum value: 1. ■ Timeout (s): the timeoutSeconds field in the YAML file. This field specifies the time (in seconds) after which a probe times out. Default value: 1. Minimum value: 1. ■ Healthy Threshold: the minimum number of times that an unhealthy container must consecutively pass health checks before it is considered healthy. Default value: 1. Minimum value: 1. For liveness probes, this parameter must be set to 1. ■ Unhealthy Threshold: the minimum number of times that a healthy container must consecutively fail health checks before it is considered unhealthy. Default value: 3. Minimum value: 1.

- Lifecycle

You can set the following parameters to configure the lifecycle of the container: Start, Post Start, and Pre Stop. For more information, see Configure the lifecycle of a container.

- **Start**: Set the command and parameter that take effect before the container starts.
- **Post Start**: Set the command that takes effect after the container starts.
- **Pre Stop**: Set the command that takes effect before the container stops.

- (Optional)Volume

  You can mount local volumes and persistent volume claims (PVCs) to the container.

  - **Add Local Storage**: You can select HostPath, ConfigMap, Secret, and EmptyDir. The source directory or file is mounted to a path in the container. For more information, see Volumes.
  - **Add PVC**: Cloud Storage is supported.

  In this example, a PVC named disk-ssd is mounted to the */tmp* path of the container.

- (Optional)Log

  Configure **Log Service**. You can specify log collection configurations and add custom tags.

  > **Notice** Make sure that the Log Service agent is installed in the cluster.

Parameter	Description
Collection Configuration	Logstore: creates a Logstore in Log Service to store collected log data.
	Log Path in Container: specifies stdout or a path to collect log data ■ stdout: specifies that the stdout files are collected. ■ Text Logs: specifies that log data in the specified path of the container is collected. In this example, */var/log/nginx* is specified as the path. Wildcard characters can be used to specify the path.
Custom Tag	You can also add custom tags. Custom tags are added to the log data of the container when the log data is collected. Log data with tags is easier to aggregate and filter.

8. Configure the parameters based on your business requirements and click **Next**.
9. (Optional)Configure advanced settings.
   - Access Control

> **Note**
>
> You can configure the following access control settings based on your business requirements:
>
> - Internal applications: For applications that run inside the cluster, you can create a ClusterIP or NodePort Service to enable internal communication.
> - External applications: For applications that are open to the Internet, you can configure access control by using one of the following methods:
>   - Create a LoadBalancer Service and enable access to your application over the Internet by using a Server Load Balancer (SLB) instance.
>   - Create an Ingress and use the Ingress to expose your application to the Internet. For more information, see Ingress.

You can also specify how the backend pods are exposed to the Internet. In this example, a ClusterIP Service and an Ingress are created to expose the NGINX application to the Internet.

Parameter	Description
Services	Click **Create** on the right side of **Services**. In the Create Service dialog box, set the parameters. Select **Cluster IP**.
Ingresses	Click **Create** on the right side of **Ingresses**. In the Create dialog box, set the parameters. **Note** When you create an application from an image, you can create an Ingress only for one Service. In this example, a virtual hostname is used as the test domain name. You must add the following entry to the hosts file to map the domain name to the IP address of the Ingress. In actual scenarios, use a domain name that has obtained an Internet Content Provider (ICP) number. `101.37.22*.***  foo.bar.com  #The IP address of the Ingress.`

You can find the created Service and Ingress in the **Access Control** section. You can click **Update** or **Delete** to change the configurations.

- Scaling

  Specify whether to enable **HPA** to automatically scale the number of pods based on the CPU and memory usage. This enables the application to run smoothly at different load levels.

  > **Note** To enable HPA, you must configure required resources for the container. Otherwise, HPA does not take effect.

  - **Metric**: Select CPU Usage or Memory Usage. The selected resource type must be the same as the one you have specified in the Required Resources field.

- **Condition**: Specify the resource usage threshold. HPA triggers scaling events when the threshold is exceeded.
- **Max. Replicas**: Specify the maximum number of replicated pods to which the application can be scaled.
- **Min. Replicas**: Specify the minimum number of replicated pods that must run.

- Scheduling

  You can set the following parameters: Update Method, Node Affinity, Pod Affinity, Pod Anti Affinity, and Toleration. For more information, see Affinity and anti-affinity.

  > **Note** Node affinity and pod affinity affect pod scheduling based on node labels and pod labels. You can add node labels and pod labels that are provided by Kubernetes to configure node affinity and pod affinity. You can also add custom labels to nodes and pods, and then configure node affinity and pod affinity based on these custom labels.

Parameter	Description
Update Method	Select Rolling Update or OnDelete. For more information, see Deployments.
Node Affinity	Set **Node Affinity** by adding labels to worker nodes. Node Affinity supports required and preferred rules, and various operators, such as In, NotIn, Exists, DoesNotExist, Gt, and Lt. ■ **Required**: Specify the rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution field of the nodeAffinity parameter. These rules have the same effect as the `NodeSelector` parameter. In this example, pods can be scheduled only to nodes with the specified labels. You can create multiple required rules. However, only one of them must be met. ■ **Preferred**: Specify the rules that are not required to be matched for pod scheduling. Pods are scheduled to a node that matches the preferred rules when multiple nodes match the required rules. In the YAML file, these rules are defined by the preferredDuringSchedulingIgnoredDuringExecution field of the nodeAffinity parameter. In this example, the scheduler attempts to schedule a pod to a node that matches the preferred rules. You can also set weights for preferred rules. If multiple nodes match the rule, the node with the highest weight is preferred. You can create multiple preferred rules. However, all of them must be met before the pod can be scheduled.
	Pod affinity rules specify how pods are deployed relative to other

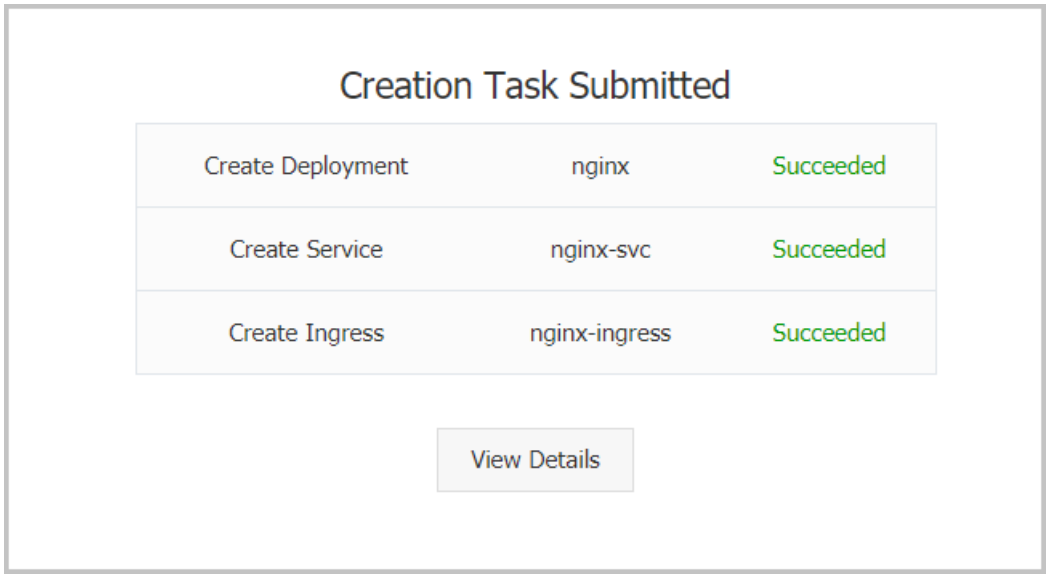
Parameter	Description
Pod Affinity	pods in the same topology domain. For example, you can use pod affinity to deploy services that communicate with each other to the same topological domain, such as a host. This reduces the network latency between these services. Pod affinity enables you to select nodes to which pods can be scheduled based on the labels of other running pods. Pod affinity supports required and preferred rules, and the following operators: `In, NotIn, Exists, and DoesNotExist`. ■ **Required**: Specify rules that must be matched for pod scheduling. In the YAML file, these rules are defined by the requiredDuringSchedulingIgnoredDuringExecution field of the podAffinity parameter. A node must match the required rules before pods can be scheduled to the node. ■ **Namespace**: Specify the namespace to apply the required rule. Pod affinity rules are defined based on the labels that are added to pods and therefore must be scoped to a namespace. ■ **Topological Domain**: Set the topologyKey. This specifies the key for the node label that the system uses to denote the topological domain. For example, if you set the parameter to `kubernetes.io/hostname`, topologies are determined by nodes. If you set the parameter to `beta.kubernetes.io/os`, topologies are determined by the operating systems of nodes. ■ **Selector**: Click Add to add pod labels. ■ **View Applications**: Click **View Applications** and set the namespace and application in the dialog box that appears. You can view the pod labels on the selected application and add the labels as selectors. ■ Required Rules: Specify labels on existing applications, the operator, and the label value. In this example, the required rule specifies that the application to be created is scheduled to a host that runs applications with the `app:nginx` label. ■ **Preferred**: Specify rules that are not required to be matched for pod scheduling. In the YAML file, preferred rules are defined by the preferredDuringSchedulingIgnoredDuringExecution field of the podAffinity parameter. The scheduler attempts to schedule the pod to a node that matches the preferred rules. You can set weights for preferred rules. The other parameters are the same as those of required rules. **Note** **Weight**: Set the weight of a preferred rule to a value from 1 to 100. The scheduler calculates the weight of each node that meets the preferred rule based on an algorithm, and then schedules the pod to the node with the highest weight.

Parameter	Description
Pod Anti Affinity	Pod anti-affinity rules specify that pods are not scheduled to topological domains where pods with matching labels are deployed. Pod anti-affinity rules apply to the following scenarios: ■ Schedule the pods of an application to different topological domains, such as multiple hosts. This allows you to enhance the stability of the service. ■ Grant a pod exclusive access to a node. This enables resource isolation and ensures that no other pod can share the resources of the specified node. ■ Schedule pods of an application to different hosts if the pods may interfere with each other. **Note** The parameters of pod anti-affinity rules are the same as those of pod affinity rules. You can create the rules for different scenarios.
Toleration	Configure toleration rules to allow pods to be scheduled to nodes with matching taints.
Schedule to Virtual Nodes	Specify whether to schedule pods to virtual nodes. This option is unavailable if the cluster does not contain a virtual node.

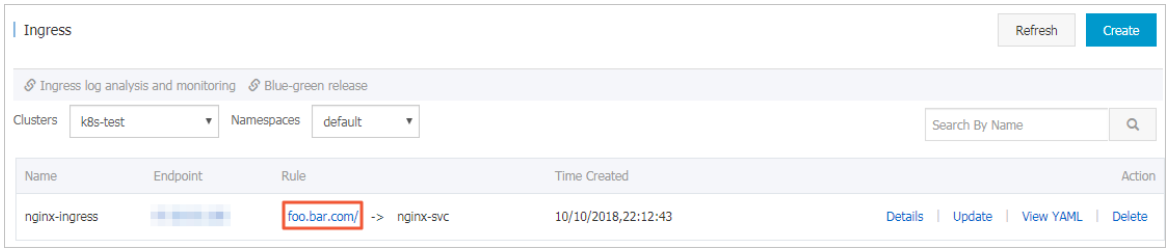
- Labels and Annotations
  - Pod Labels: Add a label to the pod. The label is used to identify the application.
  - Pod Annotations: Add an annotation to the pod.

10. Click **Create**.

After the application is deployed, you are redirected to the Complete wizard page. The resource objects of the application are displayed. You can find the resource objects under the application and click **View Details** to view application details.



11. In the left-side navigation pane, choose **Ingresses and Load Balancing > Ingresses**. The created Ingress rule is displayed on the page.



Result

Enter the test domain in the address bar of your browser and press Enter. The NGINX welcome page appears.

6.15.5. Configure a Kubernetes cluster that runs both sandboxed and Docker containers

Node pools support multiple types of container runtime. However, nodes in the same node pool must use the same type of container runtime. Container Service allows you to create node pools of different container runtime types for a cluster. This topic describes how to create a node pool that runs sandboxed containers and a node pool that runs Docker containers for a Kubernetes cluster.

Prerequisites

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

> **Notice** The Kubernetes cluster must meet the following requirements:
>
> - The cluster version must be 1.14.6-aliyun.1 or later.
> - The network plug-in must be Flannel or Terway. Terway must run in One ENI for Multi-Pod mode.
> - The volume plug-in must be CSI-Plugin 1.14.8.39-0d749258-aliyun or later. Flexvolume is not supported.
> - The logtail-ds version must be 0.16.34.2-f6647154-aliyun or later.

Considerations

- By default, a cluster can contain at most 100 nodes.
- Before you add an existing Elastic Compute Service (ECS) instance that is deployed in a virtual private cloud (VPC), make sure that an elastic IP address (EIP) is associated with the ECS instance, or a NAT gateway is created in the VPC. In addition, the nodes that you want to add to the node pool must have access to the Internet. Otherwise, the ECS instance cannot be added.

Create a node pool that runs Docker containers

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. On the **Node Pools** page, click **Create Node Pool** and set the parameters.

   For more information, see Create a Kubernetes cluster. The following table describes the parameters.

Parameter	Description
Name	Enter a name for the node pool.
Container Runtime	Select Docker. This specifies that all containers in the node pool are Docker containers.
Quantity	Specify the initial number of nodes in the node pool. If you do not need to create nodes, set this parameter to 0.
Operating System	The CentOS and Aliyun Linux operating systems are supported.

Parameter	Description
ECS Label	You can add labels to the ECS instances.
Node Label	You can add labels to the nodes in the cluster.
Custom Resource Group	Specify the resource group of the nodes to be added to the node pool.
Custom Security Group	Select a custom security group.

6. Click **OK**.

Create a node pool that runs sandboxed containers

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. On the **Node Pools** page, click **Create Node Pool** and set the parameters.

   For more information, see Create a Kubernetes cluster. The following table describes the parameters.

Parameter	Description
Name	Enter a name for the node pool.
Container Runtime	Select Sandboxed-Container. This specifies that all containers in the node pool are sandboxed containers.
Quantity	Specify the initial number of nodes in the node pool. If you do not need to create nodes, set this parameter to 0.
Operating System	Select an operating system. Sandboxed containers support only the Aliyun Linux operating system.
Mount Data Disk	You must mount a disk of at least 200 GiB.
ECS Label	You can add labels to the ECS instances.
Node Label	You can add labels to the nodes in the cluster.
Custom Resource Group	Specify the resource group of the nodes to be added to the node pool.
Custom Security Group	Select a custom security group.

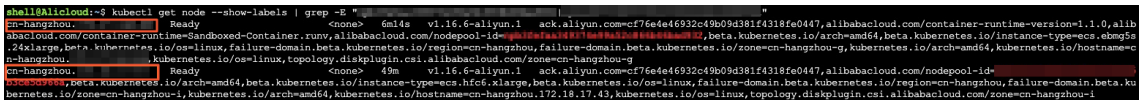
6. Click **OK**.

Result

- After you perform the preceding steps, check the states of the node pools on the **Node Pools** page. If the node pools are in the **Activate** state, the node pools are created.
- You can connect to the Kubernetes cluster and view detailed information about the nodes in the node pools.
  i. On the **Node Pools** page, select a node pool that you have created and click its name. In the **Node Pool Information** section, find and record the **Node Pool ID**.

  ii. Connect to the Kubernetes cluster by using kubectl. For more information, see Connect to a cluster through kubectl.
  iii. Run the following command to query the names of the nodes in a specified node pool:

```
kubectl get node --show-labels | grep -E "${node pool ID}|${node pool ID}"
```



  iv. Run the following command to query detailed information about a specified node:

```
kubectl get node -o wide | grep -E "${node name} {node name}"
```

6.15.6. How do I select between Docker and Sandboxed-Container?

Containers and images have become the industry standards for software packaging and delivery. Kubernetes has become a standard platform for building, developing, and managing containerized cloud-native applications. An increasing number of enterprises and customers choose to deploy their applications in Container Service. Container Service supports two types of runtime: Docker and Sandboxed-Container. This topic describes the differences between these runtimes in the following aspects: implementations and limits, commonly used commands provided by Docker Engine and Containerd, and deployment architectures. This provides references for you to select between Docker and Sandboxed-Container based on your requirements.

Differences between Docker and Sandboxed-Container in terms of implementations and limits

Item	Docker	Sandboxed-Container V2	Description
Cluster type	All types	All types	N/A
Node type	• ECS • EBM	EBM	N/A

Item	Docker	Sandboxed-Container V2	Description
Node operating system	• CentOS • Alibaba Cloud Linux2	Alibaba Cloud Linux2	• You cannot deploy both Docker and Sandboxed-Container on the same node. • To deploy both Docker and Sandboxed-Container in a cluster, you can create node pools of different runtime types.
Container engine	Docker	Containerd	N/A
Monitoring	Supported	Supported	N/A
Container log collection	Supported	Sidecar: supported. Manual configuration is required.	N/A
Container stdout collection	Supported	Supported	N/A
RuntimeClass	Not supported	Supported (runV)	N/A
Pod scheduling	No configuration is required.	• For Kubernetes V1.14.x, you must add the following configuration to the nodeSelector field: `alibabacloud.com/sandboxed-container: Sandboxed-Container.runv` • For Kubernetes V1.16.x and later, no configuration is required.	N/A
HostNetwork	Supported	Not supported	N/A
exec/logs	Supported	Supported	N/A
Node data disk	N/A	Required. The data disk must be at least 200 GiB.	N/A

Item	Docker	Sandboxed-Container V2	Description
Network plug-in	• Flannel • Terway	• Flannel • Terway: supports only the One ENI for Multi-Pod mode.	N/A
Kube-proxy mode	• Iptables • IPVS	• Iptables • IPVS	N/A
Volume plug-in	CSI Plugin	CSI Plugin	N/A
Container root file system	OverlayFS	VirtioFS	N/A

Differences in the commonly used commands provided by Docker Engine and Containerd

Docker uses Docker Engine for container lifecycle management. Sandboxed-Container uses Containerd for container lifecycle management. These tools support different commands that can be used to manage images and containers. The following table lists the commonly used commands.

Command	Docker	Containerd	
	docker	crictl (recommended)	ctr
Queries containers	docker ps	crictl ps	ctr -n k8s.io c ls
Queries container details	docker inspect	crictl inspect	ctr -n k8s.io c info
Queries container logs	docker logs	crictl logs	N/A
Runs a command in a container	docker exec	crictl exec	N/A
Mounts local standard input, output, and error streams to a running container	docker attach	crictl attach	N/A
Queries resource usage statistics	docker stats	crictl stats	N/A
Creates a container	docker create	crictl create	ctr -n k8s.io c create
Starts one or more containers	docker start	crictl start	ctr -n k8s.io run
Stops one or more containers	docker stop	crictl stop	N/A

Command	Docker	Containerd	
	docker	crictl (recommended)	ctr
Removes one or more containers	docker rm	crictl rm	ctr -n k8s.io c del
Queries images	docker images	crictl images	ctr -n k8s.io i ls
Queries image details	docker inspect	crictl inspecti	N/A
Pulls an image	docker pull	crictl pull	ctr -n k8s.io i pull
Pushes an image	docker push	N/A	ctr -n k8s.io i push
Removes one or more images	docker rmi	crictl rmi	ctr -n k8s.io i rm
Queries pods	N/A	crictl pods	N/A
Queries pod details	N/A	crictl inspectp	N/A
Starts one or more pods	N/A	crictl runp	N/A
Stops one or more pods	N/A	crictl stopp	N/A

Differences between Docker and Sandboxed-Container in terms of deployment architectures

Runtime	Deployment architecture
Docker	`kubelet -> dockerd -> containerd -> containerd-shim -> runC containers`

Sandboxed-Container V1	`kubelet -> (CRI)containerd`
	`\-> containerd-shim -> runC containers`
	`\-> containerd-shim-kata-v2 -> runV sandboxed containers`

Sandboxed-Container V2	`kubelet -> (CRI)containerd`
	`\-> containerd-shim -> runC containers`
	`\-> containerd-shim-rund-v2 -> runV sandboxed containers`

6.15.7. Benefits of Sandboxed-Container

This topic describes the advantages and application scenarios of Sandboxed-Container and provides a comparison between Sandboxed-Container and open source Kata Containers. This allows you to learn more about the benefits of Sandboxed-Container.

Context

Sandboxed-Container provides an alternative to the Docker runtime environment. It supports the following features:

- Sandboxed-Container allows your applications to run in a sandboxed and lightweight virtual machine. This virtual machine is equipped with a dedicated kernel and provides better isolation and enhanced security.
- Compared with open source Kata Containers, Sandboxed-Container is optimized for storage, networking, and stability.
- You can use Sandboxed-Container to isolate untrusted applications and applications of different tenants for higher security. You can also use Sandboxed-Container to isolate applications with faults and applications with degraded performance. This minimizes the negative impact on your service. In addition, Sandboxed-Container offers the same user experience as Docker in terms of logging, monitoring, and elastic scaling.

Benefits

Compared with Docker, Sandboxed-Container has the following benefits:

- Strong isolation based on sandboxed and lightweight virtual machines.
- Compatibility with runC in terms of application management.
- High performance that corresponds to 90% performance of applications based on runC.
- The same user experience as runC in terms of logging, monitoring, and storage.
- Support for RuntimeClass.
- Easy to use with limited expertise that is required to use virtual machines.
- Higher stability than that provided by Kata Containers.

Comparison between Sandboxed-Container and Kata Containers

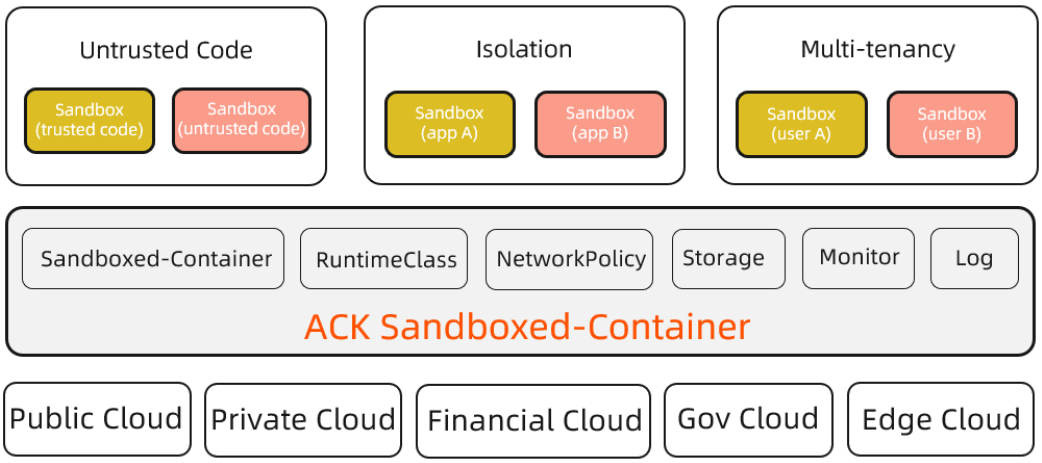
Sandboxed-Container outperforms Kata Containers in the following aspects.

Item	Category	Sandboxed-Container	Kata Containers
Sandbox startup time consumption		About 150 ms	About 500 ms
Root file system		OverlayFS over virtio-fs. Performance: ☆☆☆☆	OverlayFS over 9pfs. Performance: ☆☆
	HostPath	Disks are mounted to Sandboxed-Container over 9pfs. Performance: ☆☆	Disks are mounted to Kata Containers over 9pfs. Performance: ☆☆

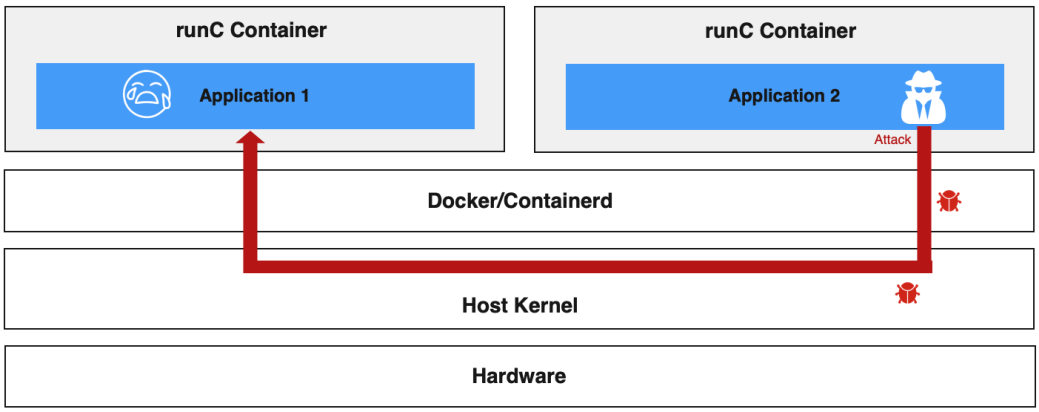
Item	Category	Sandboxed-Container	Kata Containers
Volume	EmptyDir	over VirtioFS	By default, the volume is mounted to Kata Containers over 9pfs.
	Disk	By default, cloud disks are mounted to Sandboxed-Container over virtio-fs. Performance: ☆☆☆☆	Cloud disks are mounted to Kata Containers over 9pfs. Performance: ☆☆
	NAS	By default, Apsara File Storage NAS (NAS) file systems are mounted to Sandboxed-Container over virtio-fs. Performance: ☆☆☆☆	NAS file systems are mounted to Kata Containers over 9pfs. Performance: ☆
Network plug-in		• The Terway network plug-in is used. Its network performance is 20% to 30% higher than Flannel. Terway supports features such as NetworkPolicy. This allows you to define the networking policies for pods. • Flannel	Flannel
Monitoring and alerting		• Enhanced monitoring of disks and network conditions for pods that host Sandboxed-Container. • Integrated with Cloud Monitor. This facilitates cluster monitoring and alerting.	Monitoring of disks and network conditions is unavailable for pods that host Sandboxed-Container.
Stability		☆☆☆☆☆	☆☆

Applicable scenarios of Sandboxed-Container

This section describes the applicable scenarios of Sandboxed-Container.



- Scenario 1: Sandboxed-Container can run untrusted code and applications in isolated containers. This is not supported by containers in runC.
  - Security risks of runC



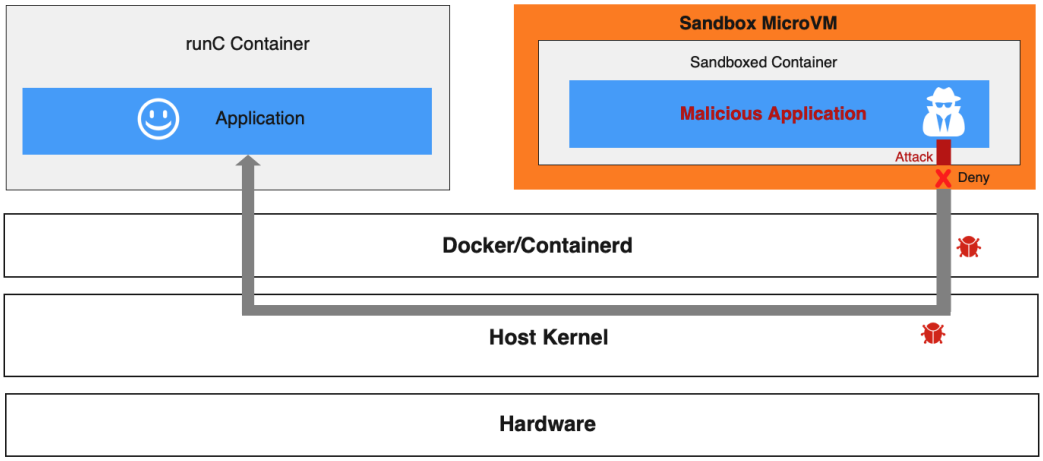
    - runC isolates containers by using namespaces and control groups (cgroups). This exposes containers to security threats.
    - All containers on a node share the host kernel. If a kernel vulnerability is exposed, malicious code may escape to the host and then infiltrate the backend network. Malicious code execution may cause privilege escalation, compromise sensitive data, and destroy system services and other applications.
    - Attackers may also exploit application vulnerabilities to infiltrate the internal network.

    You can implement the following measures to reduce security risks of containers in runC.

    - Seccomp: filters system calls.
    - SElinux: restricts the permissions of container processes, files, and users.
    - Capability: limits the capability of container processes.
    - dockerd rootless mode: forbids users to use root permissions to run the Docker daemon and containers.

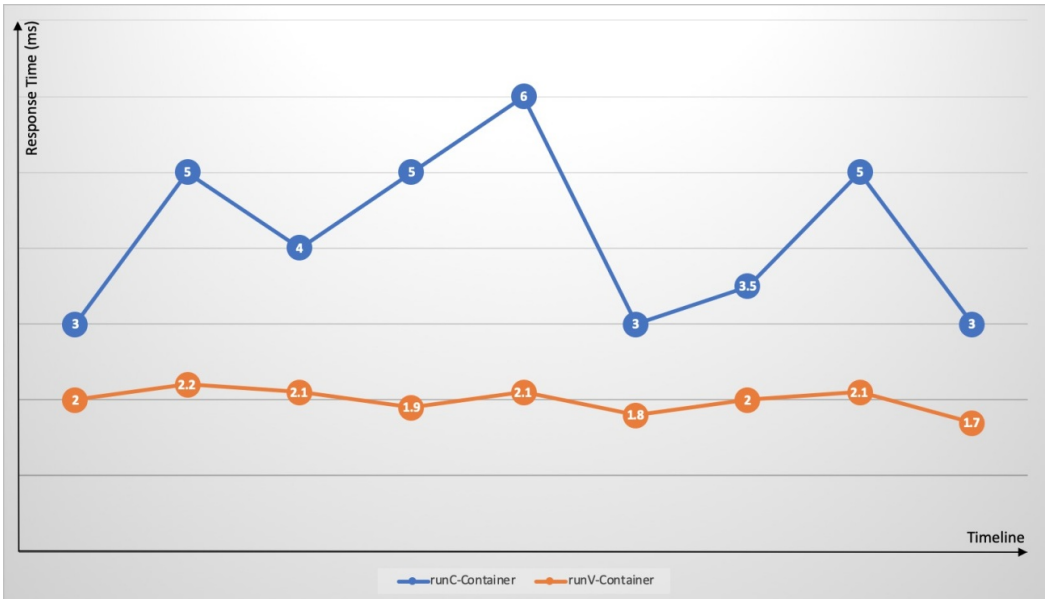
    The preceding measures can enhance the security of containers in runC and reduce attacks on the host kernel by malicious containers. However, container escapes and host kernel vulnerabilities remain unresolved.

- Sandboxed-Container prevents potential risks based on container isolation



In a Sandboxed-Container runtime environment, applications that have potential security risks are deployed on sandboxed and lightweight virtual machines. Each of the virtual machines has a dedicated guest OS kernel. If a security vulnerability is detected on a guest OS kernel, the attack is limited to one sandbox and does not affect the host kernel or other containers. The Terway network plug-in allows you to define networking policies for pods. This enables system isolation, data isolation, and network isolation for Sandboxed-Containers.

- Scenario 2: Sandboxed-Container resolves common issues of runC containers, such as fault spreading, resource contention, and performance interference.

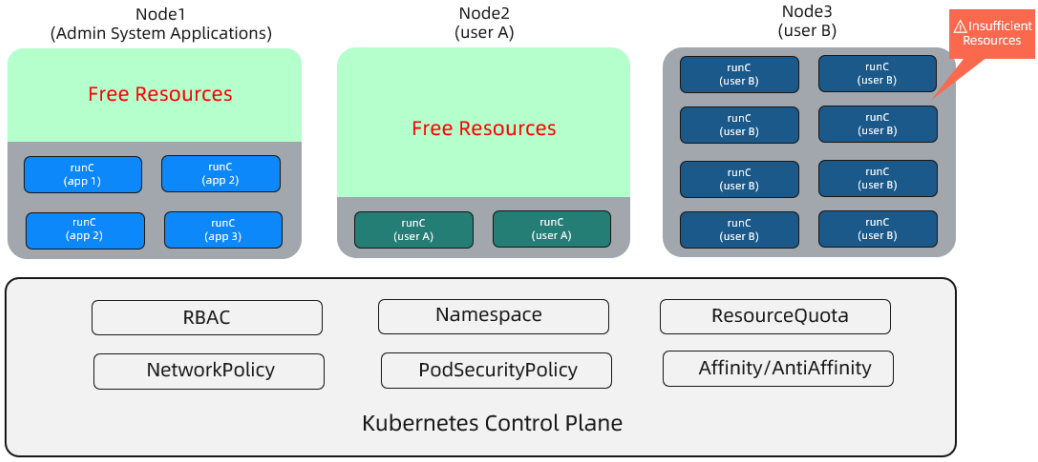


Kubernetes provides easy deployment of different containers on a single node. However, cgroups are not optimized to address resource contention. Resource-intensive applications (such as CPU-intensive, I/O-intensive applications) may compete for the same resources. This causes significant fluctuations in response time and increases the overall response time. Exceptions or faults on an application may spread to the hosting node and disrupt the running of the total cluster. For example, memory leaks and frequent core dumps of an application may overload the node, and exceptions on a container may trigger a host kernel bug that results in complete system failure. Sandboxed-Container addresses the issues that are common with runC containers by using dedicated guest OS kernels and hypervisors. The issues include failure spreading, resource contention, and performance interference.

- Scenario 3: Sandboxed-Container supports multi-tenant services.

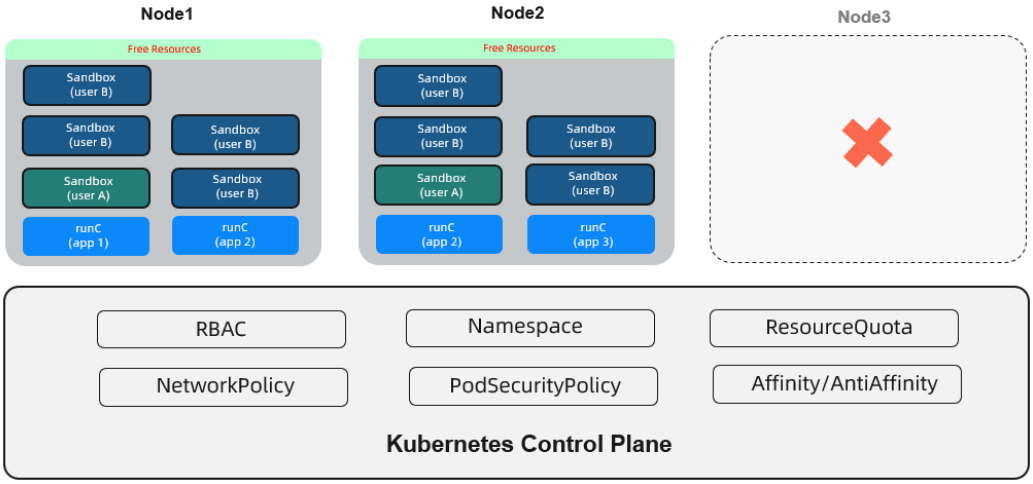
  You may need to isolate the applications of an enterprise that consists of multiple business lines or departments. For example, a financial department requires high security applications. However, other non-security-sensitive applications do not have high security requirements. Containers in runC fail to eliminate the potential risks that arise in untrusted applications. In this scenario, you can implement the following counter measures:

  - Deploy multiple independent single-tenant clusters. For example, deploy financial business and other non-security-sensitive business in different clusters.
  - Deploy a multi-tenant cluster and separate applications of different business lines by namespaces. The resource of a node is exclusive to a single business line. This solution provides data isolation for coordination with the resource quotas and network policies to implement multi-tenant isolation. Compared with multiple single-tenant clusters, this solution focuses on fewer management planes and thus reduces management costs. However, this solution cannot avoid resource waste on nodes. This issue is caused by low resource utilization of some tenants.



  Sandboxed-Container allows you to isolate untrusted applications by using sandboxed virtual machines. This prevents the risks of container escapes. This also allows you to deploy different containerized applications on each node. This way, the following benefits are provided:

  - Resource scheduling is simplified.
  - A node is no longer exclusive to a service. This improves node resource usage and reduces resource fragments and cluster resource costs.
  - Sandboxed containers use lightweight virtual machines to provide almost the same performance as containers in runC.



6.15.8. Differences between runC and runV

This topic describes the differences between runC and Sandboxed-Container (runV) in terms of their performance and pod creation methods. This allows you to better understand and utilize the benefits of sandboxed containers.

Differences between runC and runV

Item	runC	runV
Container engine	Docker and Containerd	Containerd
Node type	Elastic Compute Service (ECS) instances and ECS Bare Metal instances	EBM
Container kernel	Share the host kernel	Dedicated kernel
Container isolation	Cgroups and namespaces	Lightweight virtual machines (VMs)
Rootfs Graph Driver	OverlayFS	DeviceMapper
RootFS I/O throttling	Cgroups	DeviceMapper Block IO Limit **Note** Supported by only Sandboxed-Container V1.
NAS mounting	Not supported	Supported
Disk mounting	Not supported	Supported
Collection of container logs	Logtail directly collects container logs from the host.	logtail sidecar

Item	runC	runV
Pod Overhead	None	• Sandboxed-Container V1: For example, if you set memory: 512 Mi for a pod overhead, it indicates that 512 MiB of memory is allocated to the pod sandbox. Pod overhead refers to the amount of resources consumed by the pod sandbox. In this case, if you set a memory limit of 512 MiB for containers in the pod, the pod will request a total memory of 1,024 MiB. • Sandboxed-Container V2: The memory limit for a pod overhead is calculated based on the following formula: Memory for a pod overhead = 64 MiB + Requested memory of containers in a pod × 2%. If the result is greater than 512 MiB, the value is set to 512 Mi. If the result is smaller than 64 MiB, the value is set to 64 Mi.

Differences in pod creation between runC and runV

You can connect to clusters of Container Service by using kubectl. For more information, see Connect to a cluster through kubectl.

- Create a pod that uses runC
  i. (Optional)Use `runtimeClassName: runc` to set the container runtime to runC.

  > **Note** The preceding command is optional. runC is the default container runtime.

  ii. Run the following commands to create a pod that uses runC:

```
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Pod
metadata:
  name: busybox-runc
  labels:
    app: busybox-runc
spec:
  containers:
  - name: busybox
    image: registry.cn-hangzhou.aliyuncs.com/acs/busybox:v1.29.2
    command:
    - tail
    - -f
    - /dev/null
    resources:
      limits:
        cpu: 1000m
        memory: 512Mi
      requests:
        cpu: 1000m
        memory: 512Mi
EOF
```

- Create a pod that uses runV
  i. Use `runtimeClassName: runv` to set the container runtime to runV.
  ii. (Optional)Run the following command to verify that a RuntimeClass object named `runv` exists in the cluster.

```
kubectl get runtimeclass runv -o yaml
```

> **Note** A RuntimeClass object named `runv` is automatically created in a Kubernetes cluster that uses Sandboxed-Container.

  iii. Run the following command to create a pod that uses runV:

```
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Pod
metadata:
  name: busybox-runv
  labels:
    app: busybox-runv
spec:
  runtimeClassName: runv
  nodeSelector:
    alibabacloud.com/container-runtime: Sandboxed-Container.runv
  containers:
  - name: busybox
    image: registry.cn-hangzhou.aliyuncs.com/acs/busybox:v1.29.2
    command:
    - tail
    - -f
    - /dev/null
    resources:
      limits:
        cpu: 1000m
        memory: 512Mi
      requests:
        cpu: 1000m
        memory: 512Mi
EOF
```

Notice If the Kubernetes version is earlier than 1.16, add the following nodeSelector configuration:

```
nodeSelector:
    alibabacloud.com/container-runtime: Sandboxed-Container.runv
```

iv. Run the following command to query the pod that you have created: If the output is `runv`, it indicates that the pod is running in a sandbox.

```
kubectl get pod busybox-runv -o jsonpath={.spec.runtimeClassName}
```

v. Run the following command to log on to the pod and query its CPU and memory specifications:

```
kubectl exec -ti pod busybox-runv /bin/sh
/ # cat /proc/meminfo | head -n1
MemTotal:        1130692 kB
/ # cat /proc/cpuinfo | grep processor
processor    : 0
```

The output shows that the number of CPUs is not the same as that of the host. The total memory is the sum of pod memory and pod overhead. Be aware that the total memory is slightly smaller because the system also consumes some memory.

6.15.9. Compatibility notes

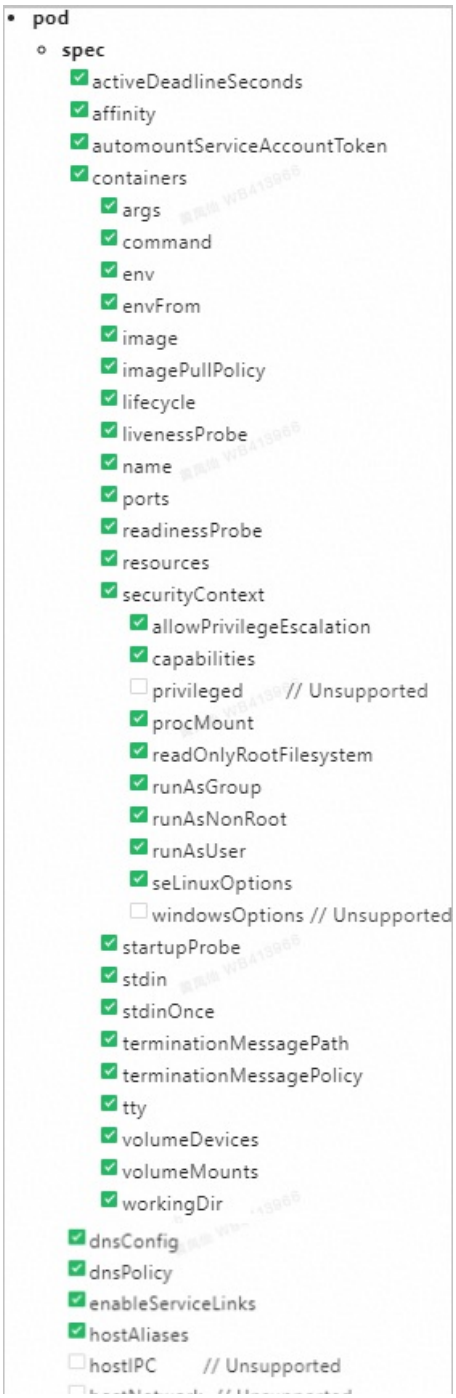
This topic describes the pod fields that are supported by Sandboxed-Container. This allows you to fully use the Sandboxed-Container runtime.

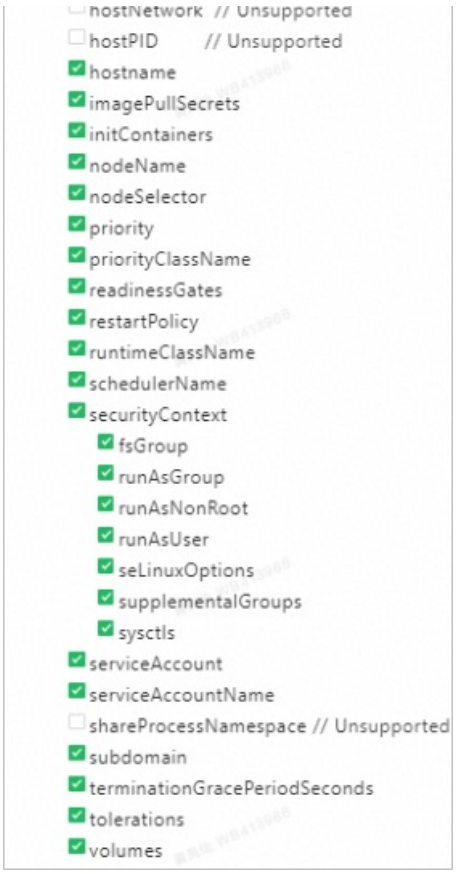
Context

Sandboxed-Container is a new runV container runtime that provides compatibility with runC in terms of pod networking, service networking (ClusterIP and NodePort), and image management. However, Sandboxed-Container does not support all pod fields. To use Sandboxed-Container, you do not need to change your development mode or image packaging method.

Supported pod fields

Sandboxed-Container supports the following pod fields that are marked by ticks:





6.16. Edge container service

6.16.1. Create an edge Kubernetes cluster

Edge Kubernetes clusters are intended for bringing cloud computing to edges (clients). Edge Kubernetes clusters can be created, managed, and maintained in the Container Service console. Container Service is a platform built on top of the edge computing infrastructure. It is also integrated with cloud computing and edge computing. This topic describes how to create an edge Kubernetes cluster.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**. On the Clusters page, click **Create Kubernetes Cluster** in the upper-right corner.
3. On the Create Cluster page, click the **Managed Edge Kubernetes** tab and set the cluster parameters.

Parameter	Description

Parameter	Description
Cluster Name	Enter a name for the cluster. The name must be 1 to 63 characters in length, and can contain digits, letters, and hyphens (-). **Note** The cluster name must be unique among clusters that belong to the same Alibaba Cloud account.
Resource Set	Select a resource set. A resource set is a container used to store resources. Each resource must belong to a resource set. After you select a resource set, virtual private clouds (VPCs) and vSwitches are filtered based on the selected resource set.
Region	Select the region where you want to deploy the cluster.
VPC	You can select a VPC from the drop-down list. ○ If the specified VPC has a NAT gateway, Container Service uses this NAT gateway. ○ If the VPC does not have a NAT gateway, the system automatically creates one. If you do not want the system to create a NAT gateway, clear **Configure SNAT for VPC**. **Note** If you disallow the system to automatically create a NAT gateway and want the VPC to access the Internet, you must manually associate the VPC with a NAT gateway or create SNAT rules for the VPC.
vSwitch	Select vSwitches. You can select up to three vSwitches that are deployed in different **zones**.
Kubernetes Version	Select a Kubernetes version.

Parameter	Description
Master Configurations	Set the Instance Type and System Disk parameters: ○ Master Node Quantity: You can add up to three master nodes. ○ Instance Type: You can select multiple instance types. For more information, see *Instance families and instance types* in *ECS*. ○ System Disk: **SSD Disk** and **Ultra Disk** are supported. **Note** You can select **Enable Backup** to back up disk data.
Worker Instance	You can select **Create Instance** or **Add Existing Instance**.
Worker Configurations	If **Worker Instance** is set to **Create Instance**, set the following parameters: ○ Instance Type: You can select multiple instance types. For more information, see *Instance families and instance types* in *ECS*. ○ Selected Types: The selected instance types are displayed. ○ Quantity: Set the number of worker nodes. By default, worker nodes are not required in edge Kubernetes clusters. Therefore, the value of this parameter is set to 0. ○ System Disk: **SSD Disk** and **Ultra Disk** are supported. **Note** You can select **Enable Backup** to back up disk data. ○ Mount Data Disk: **SSD Disk** and **Ultra Disk** are supported. **Note** ■ You can select **Encrypt Disk** to encrypt disks. ■ You can select **Enable Backup** to back up disk data.
Operating System	The CentOS and Alibaba Cloud Linux operating systems are supported.

Parameter	Description
Network Plug-in	Edge Kubernetes clusters support only Flannel. You do not need to set this parameter.
Password	Set a password that is used to log on to the nodes. **Note** The password must be 8 to 30 characters in length, and must contain at least three of the following types of character: uppercase letters, lowercase letters, digits, and special characters.
Confirm Password	Enter the password again.
Pod CIDR Block and Service CIDR	For more information, see *Network planning* in *VPC User Guide*. **Note** These parameters are available only when you select an **existing VPC**.
Configure SNAT	This parameter is optional. If you clear Configure SNAT for VPC, you must create a NAT gateway or configure SNAT rules for the VPC. **Note** For edge Kubernetes clusters, we recommend that you select Configure SNAT for VPC.

Parameter	Description
Access to the Internet	Specify whether to expose the API server with an elastic IP address (EIP). The Kubernetes API server provides multiple HTTP-based RESTful APIs that can be used to create, delete, modify, query, and watch resource objects such as pods and Services. **Note** We recommend that you expose the API server with an EIP. ○ If you select this check box, an EIP is created and attached to an internal-facing Server Load Balancer (SLB) instance. Port 6443 used by the API server is exposed on the master nodes. You can connect to and manage the cluster by using kubeconfig files over the Internet. ○ If you clear this check box, no EIP is created. You can connect to and manage the cluster by using kubeconfig files only from within the VPC.
SSH Logon	To enable SSH logon, you must first select Expose API Server with EIP. ○ If you enable SSH logon over the Internet, you can access the cluster by using SSH. ○ If you disable SSH logon over the Internet, you cannot access the cluster by using SSH or kubectl. If you want to access an Elastic Compute Service (ECS) instance in the cluster by using SSH, you must manually bind an EIP to the ECS instance and configure security group rules to open SSH port 22.
Log Service	If you enable Log Service, you can select an existing project or create a project. If you select **Enable Log Service**, the Log Service plug-in is automatically installed in the cluster. If you select **Create Ingress Dashboard**, Ingress access logs are collected and displayed on dashboards.
Deletion Protection	If you select this check box, the cluster cannot be deleted in the console or by calling API operations.
Node Protection	This check box is selected by default to prevent nodes from being deleted in the console or by calling API operations.

4. Configure the advanced settings.

Parameter	Description
IP Addresses per Node	The number of IP addresses that is assigned to a node. We recommend that you use the default value.
Custom Image	You can select a custom image. After you select a custom image, all nodes in the cluster are deployed by using this image.
Kube-proxy Mode	**iptables** and **IPVS** are supported. ○ iptables is a mature and stable kube-proxy mode. It uses iptables rules to conduct service discovery and load balancing. The performance of this mode is restricted by the size of the Kubernetes cluster. This mode is suitable for Kubernetes clusters that manage a small number of Services. ○ IPVS is a high-performance kube-proxy mode. It uses Linux Virtual Server (LVS) to conduct service discovery and load balancing. This mode is suitable for clusters that manage a large number of Services. We recommend that you use this mode in scenarios where high-performance load balancing is required.
Node Port Range	Specify the value of **Node Port Range**.
Taints	Add taints to all of the worker nodes in the cluster. We recommend that you do not add additional taints in case the system components cannot be deployed in the edge Kubernetes cluster.
CPU Policy	Specify the CPU policy. Valid values: ○ None: indicates that the default CPU affinity is used. This is the default policy. ○ Static: allows pods with specific resource characteristics on the node to be granted with enhanced CPU affinity and exclusivity.
Cluster Domain	The default domain name of the cluster is cluster.local. You can specify a custom domain name.
Cluster CA	Specify whether to enable the cluster certification authority (CA) certificate.

Parameter	Description
User Data	Customize the startup behaviors of ECS instances and import data to the ECS instances. The user data can be used to perform the following operations: ○ Run scripts during instance startup. ○ Pass user data as common data into an ECS instance for future reference.

5. Click **Create Cluster** in the upper-right corner of the page.
6. On the **Confirm** page, after all check items are verified, select the terms of service and disclaimer and click **OK** to start the deployment.

Result

After the cluster is created, you can find the cluster on the **Clusters** page in the console.

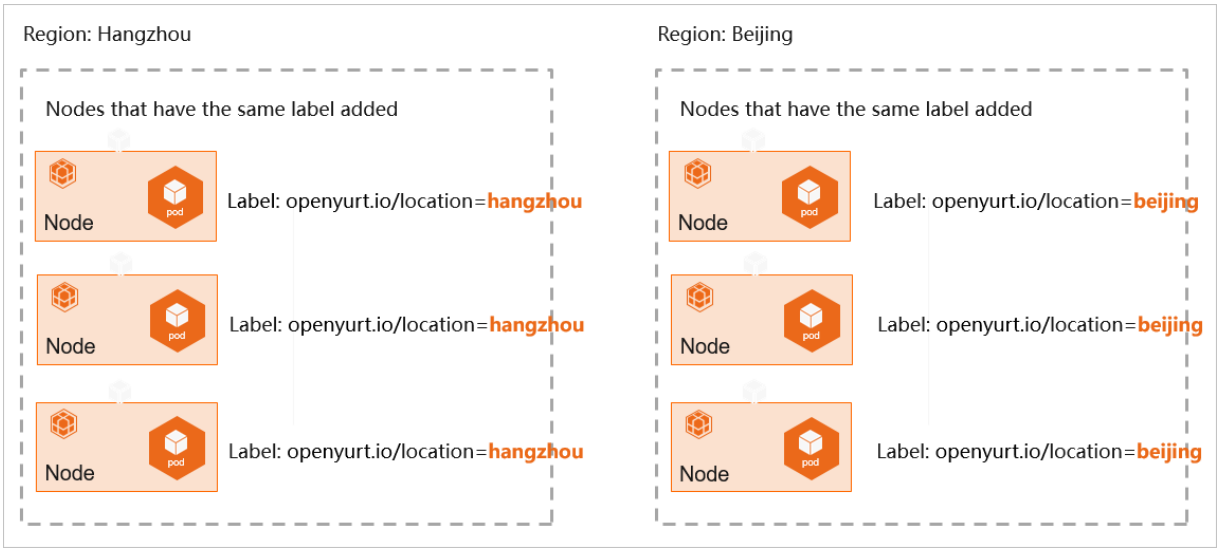
6.16.2. Edge node pools

6.16.2.1. Edge node pool overview

In edge computing scenarios, edge container service allows you to abstract nodes as edge node pools based on node attributes. This allows you to control and manage nodes in different regions in a unified manner. This topic describes edge node pools and how edge nodes are managed by edge node pools.

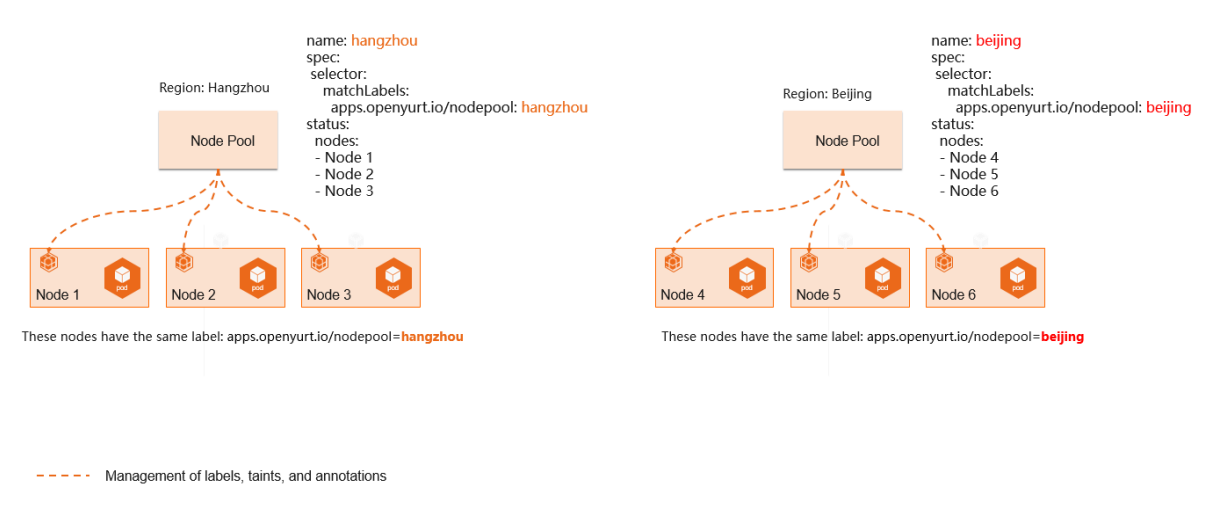
Traditional node management

In edge computing scenarios, edge nodes can be classified by different attributes such as CPU architecture, Internet service provider (ISP), and cloud service provider. Traditionally, labels are used to classify and manage nodes. However, as the numbers of nodes and labels increase, it becomes more complex to manage and maintain nodes. The following figure shows the traditional way of node management.



Edge node pools

Edge node pools allow you to classify nodes from a different dimension. You can centrally manage and maintain edge nodes that are deployed in different regions by using edge node pools, as shown in the following figure.



6.16.2.2. Create an edge node pool

An edge node pool manages a group of nodes in a cluster. For example, you can centrally manage labels and taints for the nodes in a node pool. This topic describes how to create an edge node pool in the Container Service console.

Prerequisites

- A managed edge Kubernetes cluster is created. For more information, see Create an edge Kubernetes cluster.
- The Kubernetes version of your cluster is 1.18 or later.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane of the Container Service console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click its name or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. On the **Node Pools** page, click **Create Edge Node Pool (Beta)** in the upper-right corner of the page.
6. In the **Create Edge Node Pool (Beta)** dialog box, set the parameters and click **Submit**.

Parameter	Description
Name	The name of the edge node pool.
Coordination Network between Cloud and Edge	By default, Basic is selected.

Parameter	Description
Maximum Nodes	The maximum number of nodes that can be added to the edge node pool.
Node Label	You can add labels to the nodes in the edge node pool.
Taints	You can add taints to the nodes in the edge node pool.

After the edge node pool is created, you can view information about the node pool in the node pool list.

6.16.2.3. Add nodes to an edge node pool

You can add worker nodes to an edge node pool that you created. Make sure that these worker nodes can communicate with the API server. This topic describes how to add nodes to an edge node pool.

Prerequisites

- An edge node pool is created. For more information, see Create an edge node pool.
- The Kubernetes version of your cluster is 1.18 or later.

> **Notice** You can add only nodes that run CentOS 7.4, CentOS 7.6, or Ubuntu 18.04.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane of the Container Service console, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage. Then, click the name of the cluster or click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Node Pools**.
5. On the **Node Pools** page, find the edge node pool to which you want to add nodes and click **Add Existing Node** in the **Actions** column.

   Perform the same steps as you do when you add nodes to an edge Kubernetes cluster. For more information, see Add nodes to an edge Kubernetes cluster.

   After you add nodes to the edge node pool, you can click **Details** in the **Actions** column to view the nodes that you added.

6.16.3. Edge nodes

6.16.3.1. Add nodes to an edge Kubernetes cluster

You can add worker nodes to an edge Kubernetes cluster in the Container Service console. However, you must make sure that the added nodes can communicate with the Kubernetes API server of the cluster. This topic describes how to add nodes to an edge Kubernetes cluster.

Prerequisites

Create an edge Kubernetes cluster

Context

By default, a cluster can contain at most 50 nodes.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, click the name of the cluster that you want to manage.
4. In the left-side navigation pane of the cluster details page, choose **Nodes > Nodes**. On the page that appears, click **Add Existing Node** in the upper-right corner.
5. On the **Select Existing ECS Instance** wizard page, click **Next Step**.

   You can only manually add nodes to edge Kubernetes clusters.
6. On the **Specify Instance Information** wizard page, set the parameters and click **Next Step**.

Parameter	Description	Default
flannelIface	The name of the network interface controller (NIC) that is used by the Flannel plug-in.	The name of the NIC that is specified in the default route entry of the node.
enableIptables	Specifies whether to enable iptables.	false
quiet	Specifies whether to answer all questions with yes when you add nodes.	false
manageRuntime	Specifies whether to use edgeadm to install and manage the runtime.	false
nodeNameOverride	The name of the node.	◦ "". This is the default value. This value specifies that the hostname is used as the node name. ◦ "*". This value specifies that a random string that contains six characters is used as the node name. ◦ "*."*.XXX". This value specifies that a random string that is followed by a suffix is used as the node name. The random string contains six characters.

Parameter	Description	Default
allowedClusterAddons	The list of add-ons to be installed. By default, this parameter is empty. This indicates that no add-on is installed. For a standard edge node, set this parameter to ["kube-proxy","flannel","coredns"].	[]
gpuVersion	Specifies whether the node to be added is a GPU-accelerated node. By default, this parameter is empty. Supported GPU models are Nvidia_Tesla_T4, Nvidia_Tesla_P4, and Nvidia_Tesla_P100.	"". This is the default value. This value specifies that the node to be added is not a GPU-accelerated node.
inDedicatedNetwork	Specifies whether an Express Connect circuit is used to connect to the managed edge Kubernetes cluster.	false
labels	Specifies the labels to be added to the node.	{}
annotations	Specifies the annotations to be added to the node.	{}

Parameter	Description	Default
nodeIface	This parameter specifies the following information: ○ Specifies the node IP address that kubelet retrieves from the specified network interface. If you do not specify this parameter, kubelet attempts to retrieve the node IP address in the following order: ■ Searches */etc/hosts* for the node whose name is the same as the specified hostname. ■ Finds the IP address of the network interface that is specified in the default route entry of the node. ○ Specifies the name of the NIC that is used by the Flannel plug-in. In this case, this parameter is equivalent to the flannellface parameter. This parameter will soon replace the flannellface parameter.	""

7. On the **Complete** wizard page, copy the script to the node that you want to add to the edge Kubernetes cluster and click **Done**.

8. Log on to the edge node and execute the script. This way, the node is added to the edge Kubernetes cluster.

6.16.3.2. Configure node autonomy

If an edge node is autonomous, applications run as expected on the edge node even if the edge node is disconnected from the cloud. This ensures that applications are not removed or migrated to other edge nodes in the case of network errors. This topic describes how to set the autonomy attribute for edge nodes.

Prerequisites

- Create an edge Kubernetes cluster
- Add nodes to an edge Kubernetes cluster

Context

You can enable or disable node autonomy in the Container Service console.

- If an autonomous edge node is disconnected from the cloud, Container Service does not migrate applications on this node to other nodes, and the applications are automatically restored. Node autonomy is applicable to edge computing scenarios where the network connection is weak.
- If a non-autonomous edge node is disconnected from the cloud, the node fails to send heartbeats to the nodes in the cloud. As a result, the state of the node is changed to **Not Ready** and the applications on the node are removed or migrated to other nodes after a specific time period.

Procedure

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Nodes > Nodes**.
5. On the **Nodes** page, find the node that you want to manage and choose **More > Node Autonomy Setting** in the **Actions** column.

   > **Note** The **Node Autonomy Setting** option is available only to edge nodes.

6. In the **Node Autonomy Setting** dialog box, click **OK**.

   > **Note** By default, edge nodes are not autonomous when they are added to the cluster. You can follow the preceding steps to enable or disable node autonomy.

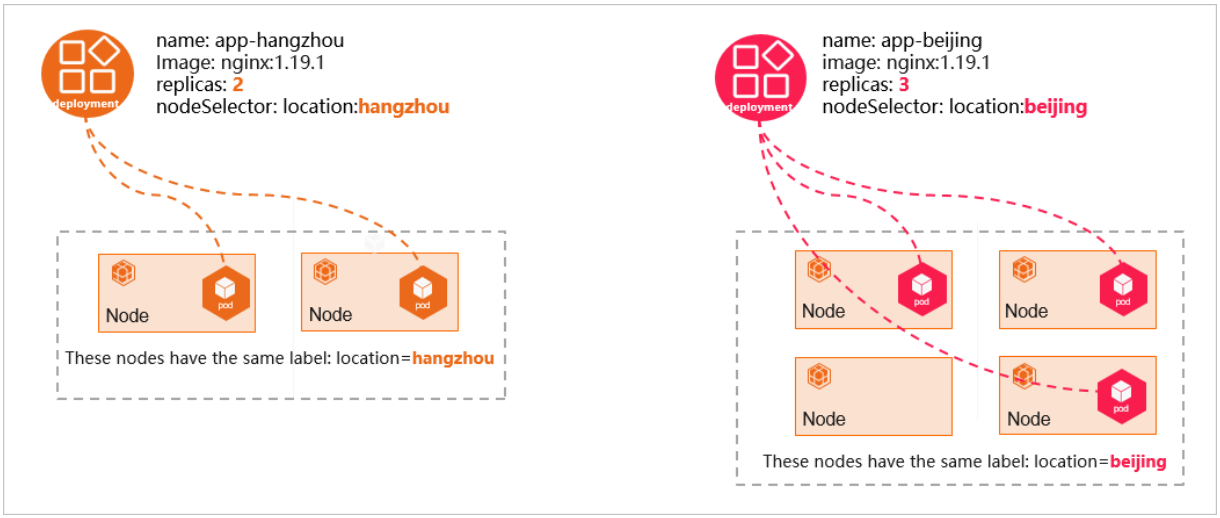
6.16.4. Cell-based management at the edge

6.16.4.1. Use the UnitedDeployment controller to deploy applications

In edge computing scenarios, you can use the UnitedDeployment controller to deploy applications to different node pools. This way, you can centrally manage the number of pods and the image version of containers by using node pools. This topic describes how to use the UnitedDeployment controller to deploy applications.

Context

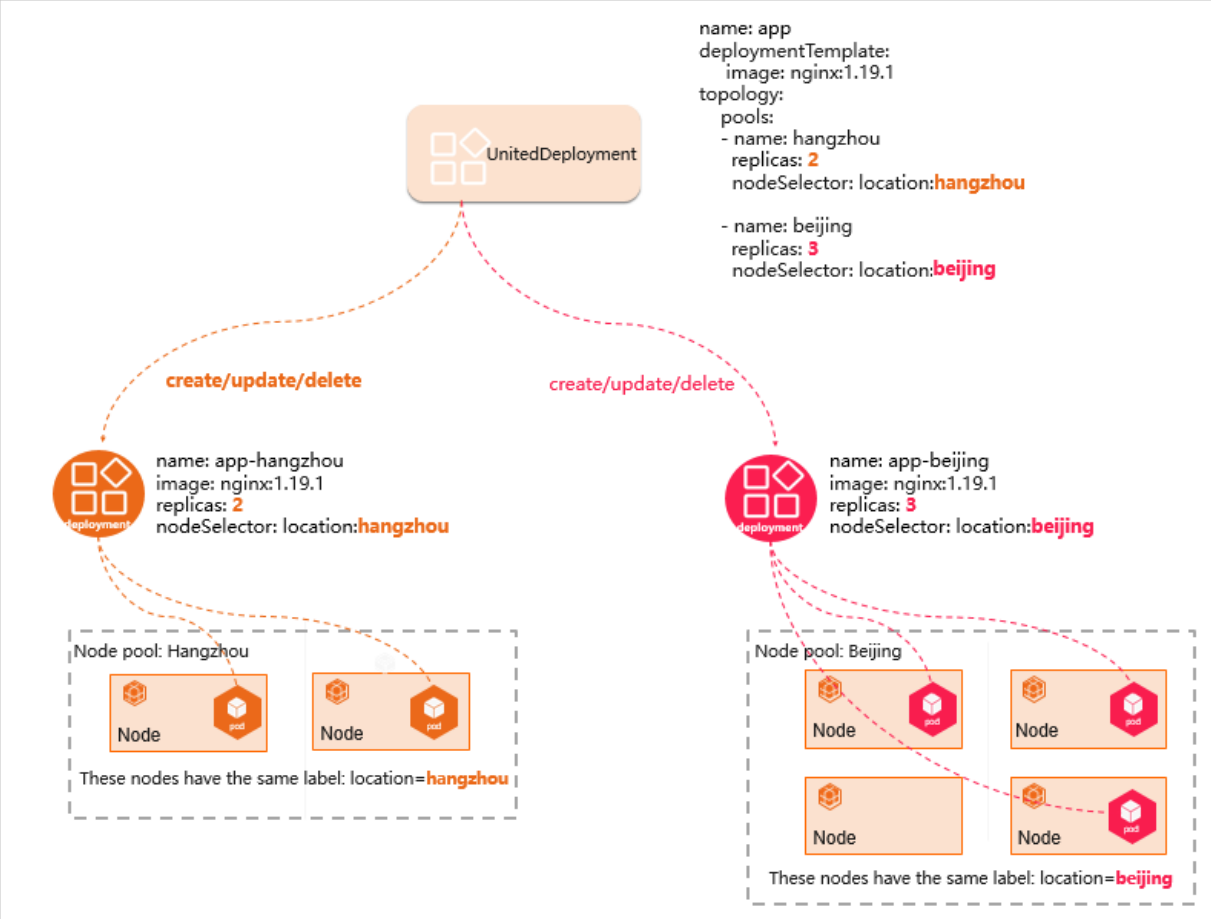
In edge computing scenarios, computing nodes may be deployed across regions, and an application may run on nodes in different regions. A Deployment is used as an example in this topic. Traditionally, you add the same label to the nodes that are deployed in the same region and create multiple Deployments. These Deployments are deployed to nodes in different regions by matching node selectors.



Application management and maintenance become more complex with the increasing number of regions and differentiated requirements for applications in different regions. The following list describes the main challenges:

- When a new image version is released, you must modify the image version for each Deployment.
- You must customize naming conventions to identify Deployments that belong to the same application.
- Deployments that belong to the same application are configured in the same way, except for the name, node selector, and number of replicated pods.

The UnitedDeployment controller is a feature provided for dedicated edge Kubernetes clusters. This feature allows you to centrally manage Deployments from a different dimension. For example, you can create, update, and delete multiple Deployments at a time.



The UnitedDeployment controller provides a template to define applications. This template allows you to deploy workloads in different regions and define the nodes in each region as a node pool. The UnitedDeployment controller supports two types of workload: StatefulSet and Deployment. The UnitedDeployment controller creates Deployments or StatefulSets based on the configurations of node pools. You can specify the number of replicated pods for each type of workload. UnitedDeployment enables automatic management and maintenance of multiple Deployments or StatefulSets within individual node pools. In addition, you can create differentiated configurations for these Deployments or StatefulSets, such as the name, node selector, and number of replicated pods.

Create a UnitedDeployment

Create a UnitedDeployment to deploy Deployments.

The following YAML template is an example:

```
apiVersion: apps.openyurt.io/v1alpha1
kind: UnitedDeployment
metadata:
  name: example
  namespace: default
spec:
  revisionHistoryLimit: 5
  selector:
    matchLabels:
      app: example
  workloadTemplate:
    deploymentTemplate:
```

```
      metadata:
        creationTimestamp: null
        labels:
          app: example
      spec:
        selector:
          matchLabels:
            app: example
        template:
          metadata:
            creationTimestamp: null
            labels:
              app: example
          spec:
            containers:
            - image: nginx:1.19.3
              imagePullPolicy: Always
              name: nginx
            dnsPolicy: ClusterFirst
            restartPolicy: Always
  topology:
    pools:
    - name: cloud
      nodeSelectorTerm:
        matchExpressions:
        - key: apps.openyurt.io/nodepool
          operator: In
          values:
          - cloud
      replicas: 2
    - name: edge
      nodeSelectorTerm:
        matchExpressions:
        - key: apps.openyurt.io/nodepool
          operator: In
          values:
          - edge
      replicas: 2
      tolerations:
      - effect: NoSchedule
        key: apps.openyurt.io/taints
        operator: Exists
```

The following table describes the fields in the YAML template.

Field	Description
spec.workloadTemplate	The workload template. Valid values: `deploymentTemplate` and `statefulSetTemplate`.
spec.topology.pools	Configurations of multiple node pools.

Field	Description
spec.topology.pools[*].name	The name of the node pool.
spec.topology.pools[*].nodeSelectorTerm	Specifies node affinity for the node pool. Set the key to `apps.openyurt.io/nodepool` and the value to the name of the node pool.
spec.topology.pools[*].tolerations	Sets tolerations for the node pool.
spec.topology.pools[*].replicas	The number of pods in each node pool.

Use the UnitedDeployment controller to manage pods

- Upgrade pods: You can modify the `spec.template.workloadTemplate.deploymentTemplate` field to trigger pod upgrades. The UnitedDeployment controller updates the workload template for all node pools. Then, the node pool controller upgrades the pods in the node pools.
- Scale the number of replicated pods for multiple node pools: You can modify the `spec.topology.pools` field to change the number of replicated pods for multiple node pools. Then, the replicated pods in the node pools are scaled based on the configuration.

6.16.4.2. Configure a Service topology

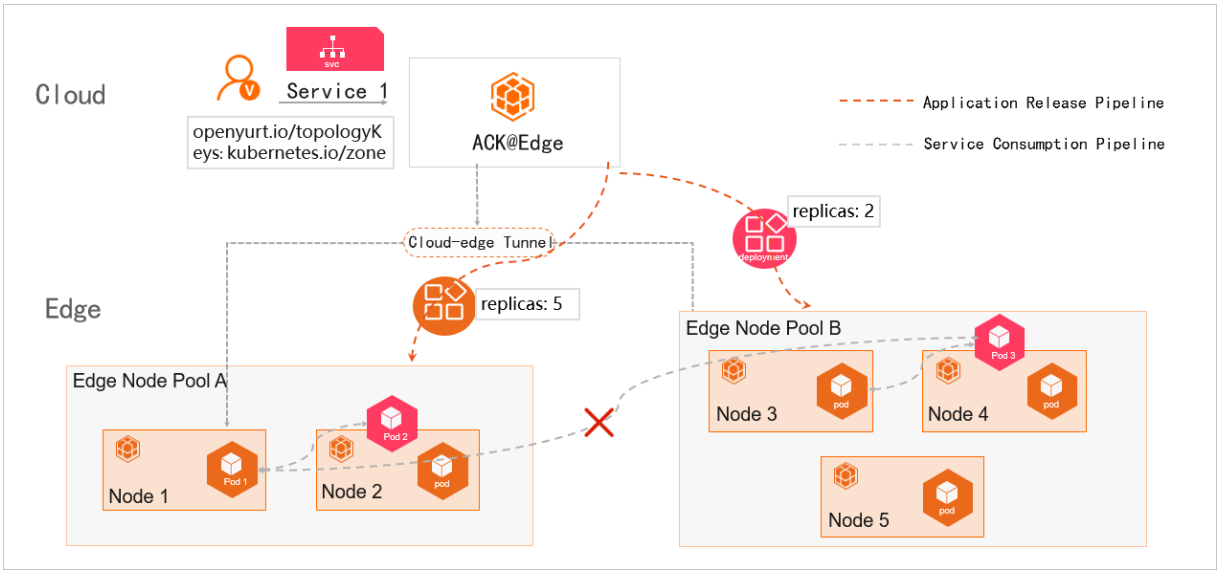
The backend endpoints of Kubernetes-native Services are randomly distributed across nodes. Consequently, when Service requests are distributed to nodes across node groups, these requests may fail to reach the nodes or may not be answered promptly. You can configure a Service topology to expose an application on an edge node only to the current node or nodes in the same edge node pool. This topic describes how a Service topology works and how to configure a Service topology.

Context

In edge computing, edge nodes are classified into groups by zone, region, and other logical attributes, such as CPU architecture, Internet service provider (ISP), or cloud service provider. Nodes in different groups are isolated from each other in one way or another. For example, these nodes may not be able to connect to each other, may not share the same resources, may have heterogeneous resources, and may run applications that are independent of each other.

How a Service topology works

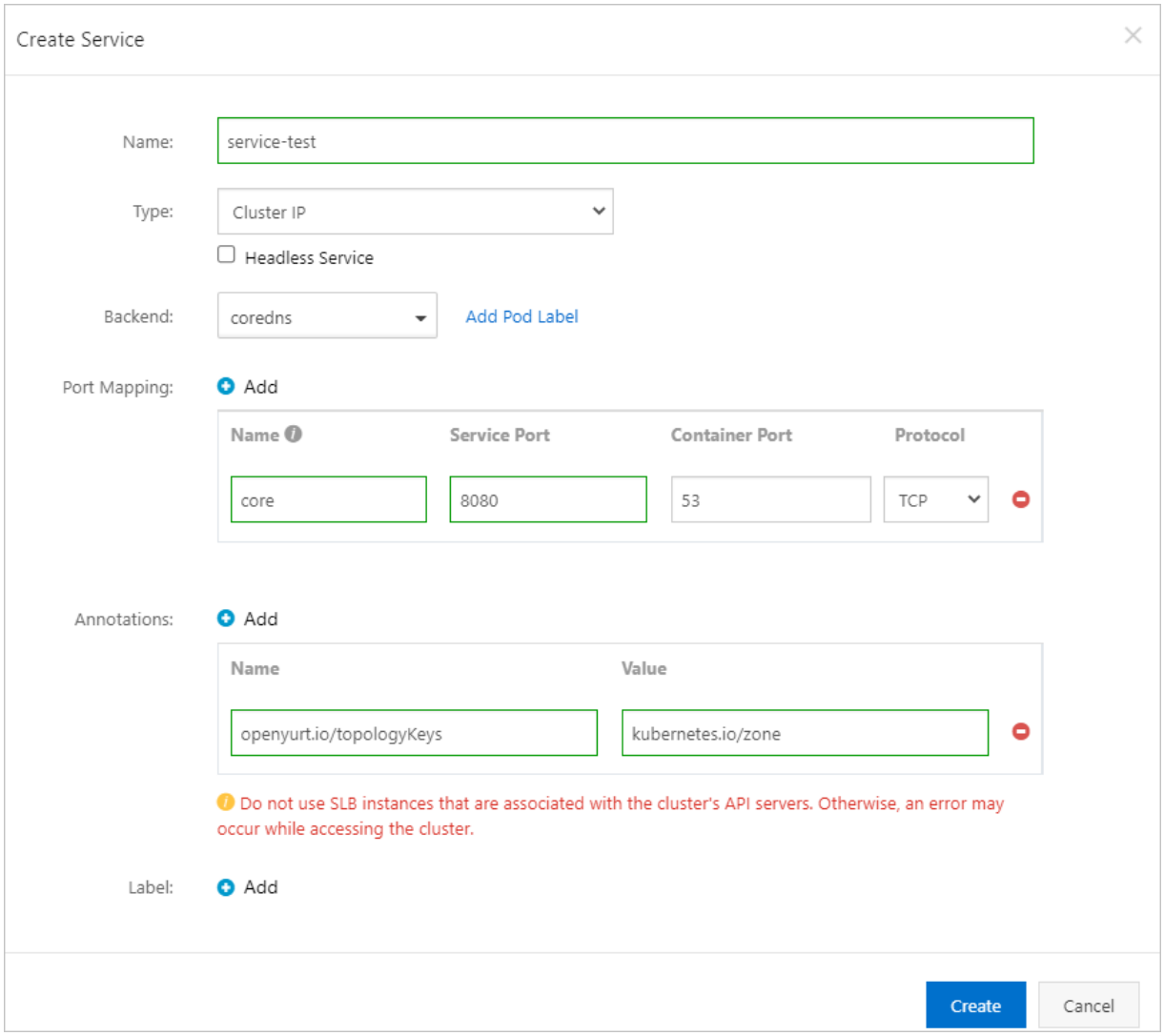
To solve the preceding issues, dedicated edge Kubernetes clusters provide a feature to manage the topology of endpoints of Kubernetes-native Services. You can configure a Service topology to specify how endpoints of a Service are accessed. For example, you can configure a Service topology to expose an application on an edge node only to the current node or nodes in the same edge node pool. The following figure shows how a Service topology works.



- Service 1 is associated with Pod 2 and Pod 3. `annotation:"openyurt.io/topologyKeys: kubernetes.io/zone"` specifies the node pool that is allowed to access Service 1.
- Pod 2 is deployed on Node 2 and Pod 3 is deployed on Node 3. Node 2 belongs to Node Pool A and Node 3 belongs to Node Pool B.
- Pod 3 and Pod 1 do not belong to the same node pool. As a result, when Pod 1 accesses Service 1, the traffic is forwarded only to Pod 2. The traffic is not forwarded to Pod 3.

Method 1: Configure a Service topology in the console

To create a Service that can be accessed only by the node pool where the Service is deployed, you only need to add an annotation to the Service. For example, you can set **Name** to `openyurt.io/topologyKeys` and **Value** to `kubernetes.io/zone`. For more information about how to create a Service, see Create a Service.



Method 2: Configure a Service topology by using a CLI

You can use a CLI to configure a Service topology in the following ways:

- Create a Service that uses the topological domain of a specific node pool. The following code block is an example of the YAML template:

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    openyurt.io/topologyKeys: kubernetes.io/zone
  name: my-service-nodepool
  namespace: default
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: nginx
  sessionAffinity: None
  type: ClusterIP
```

- Run the following command to configure the Service topology. The Service uses the topological domain of the specified node pool.

```
kubectl annotate service xxx openyurt.io/topologyKeys='kubernetes.io/zone'
```

Annotations

You can add annotations to a Kubernetes-native Service to configure a Service topology. The annotations are described in the following table.

annotation Key	annotation Value	Description
openyurt.io/topologyKeys	kubernetes.io/hostname	Specifies that the Service can be accessed only by the node where the Service is deployed.
openyurt.io/topologyKeys	kubernetes.io/zone	Specifies that the Service can be accessed only by the nodes in the node pool where the Service is deployed.
None	None	Specifies that access to the Service is unlimited.

6.16.5. Cloud-edge tunneling

By default, Container Service deploys the **edge-tunnel-server or edge-tunnel-agent** component after you create a cluster. This improves user experience. These components are used to establish tunnels from the cloud to the edge. After the tunnels are established, you can access edge nodes from the cloud. This topic describes the tunneling components provided by managed edge Kubernetes clusters and the features of these components.

Background information

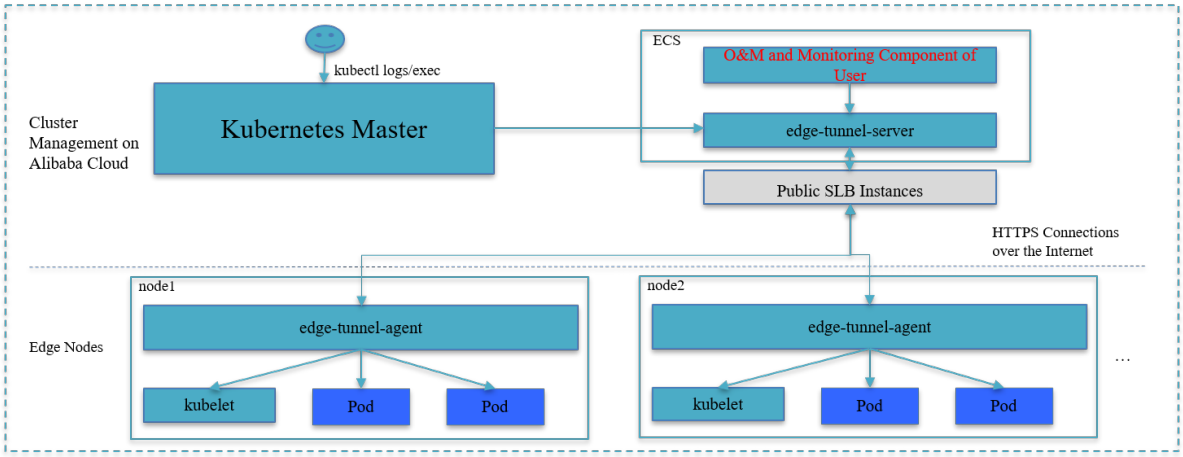
- In a Kubernetes cluster, the controller components in the cloud must run commands in kubelet to manage and maintain edge nodes. The monitoring components must retrieve monitoring data of

edge nodes from the cloud. If the edge nodes of a managed edge Kubernetes cluster are deployed in an internal network, you cannot directly access the edge nodes from the cloud.

- The **edge-tunnel-server** component is deployed as a Deployment on nodes in the cloud. The **edge-tunnel-agent** component is deployed as a DaemonSet on each edge node.

Introduction

- edge-tunnel-server is automatically installed on master nodes when you create an edge Kubernetes cluster.
- To establish secure and encrypted tunnels over the Internet, the system creates a Server Load Balancer (SLB) instance for the Service that is created by **edge-tunnel-server**. The **edge-tunnel-agent** component on each edge node establishes a tunnel to the cloud through the SLB instance. The following figure shows how cloud-edge tunneling works.



> **Note**
>
> - When edge nodes are disconnected from the cloud or the network connection is weak, the tunnels may fail to work as normal.
> - If you delete or stop the SLB instance through which the tunnels are established, the tunnels cannot work as normal.

6.17. Use the Kubernetes event center

The event center feature allows you to log Kubernetes events, query events, and configure alerting. This topic describes how to enable and view the Kubernetes event center.

Enable the Kubernetes event center

Enable the Kubernetes event center when a Kubernetes cluster is created

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**. On the page that appears, click **Create Kubernetes Cluster** in the upper-right corner.
3. On the **Create Cluster** page, select **Install node-problem-detector and Create Event Center** in the **Log Service** field. For more information about other parameters, see Create a Kubernetes cluster. Then, click **Create Cluster**.
4. On the **Confirm** dialog box, after all check items are verified, select the terms of service and

disclaimer and click **OK** to create the cluster.

Enable the Kubernetes event center in App Catalog

1. Log on to the Container Service console.
2. In the left-side navigation pane, choose **Marketplace > App Catalog**.
3. On the **App Catalog** page, find and click ack-node-problem-detector.
4. On the **App Catalog - ack-node-problem-detector** page, click the **Parameters** tab and set `eventer.sinks.sls.enabled` to `true`.

```
sinks:
  sls:
    enabled: true
    # If you want the monitoring results to be notified by sls, set enabled to true.
    topic: ""
    project:  "k8s-log-cc7640381ac7f4a99971f55a2744a2748"
    # You can view the project information by logging in to the
    # SLS console. Please fill in the name of the project here.
    # eg: your project name is k8s-log-cc18a5f3443dhdss22654da,
    # then you can fill k8s-log-cc18a5f3443dhdss22654da to project label.
    logstore: "k8s-event"
    # You can view the project information by logging in to the
    # SLS console. Please fill the logstore address in here.
```

5. On the **App Catalog - ack-node-problem-detector** page, click the **Description** tab.
6. In the **Deploy** pane, select the cluster and click **Create**.

Access the Kubernetes event center

Access the Kubernetes event center in the Container Service console

1. Log on to the Container Service console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.
4. In the left-side navigation pane of the details page, choose **Operations > Event Center**.
5. Access the Kubernetes event center on the Event Center page

> **Note** If the Event Center page does not appear, check whether the Kubernetes event center is enabled.

   - On the **Event Center** page, click the **Events Overview** tab to go to the overview page of the Kubernetes event center.
   - On the **Event Center** page, click the **Cluster Events Query** tab to customize query conditions.
   - On the **Event Center** page, click the **Pod Events** tab to query events of pods.

Access the Kubernetes event center by using Log Service

1. View the cluster ID.
   i. Log on to the Container Service console.
   ii. In the left-side navigation pane, click **Clusters**.
   iii. On the **Clusters** page, find the cluster that you want to manage and click **Details** in the **Actions** column.

iv. Click the **Basic Information** tab to view the cluster ID.

2. Log on to the Log Service console.
3. In the search box of the Projects section, enter and click k8s-log-<CLUSTER_ID>.

> **Note** Replace the CLUSTER_ID with the cluster ID that you obtained in Step .

4. In the Logstores section, choose **K8s-event > Visual Dashboards > Kubernetes Event Center V1.2**.
5. On the **Kubernetes Event Center V1.2** page, you can view the trends of WARNING events and ERROR events.